



Red Hat Enterprise Linux 7 System Administrator's Guide

Deployment, Configuration and Administration of Red Hat Enterprise Linux 7

Jaromír Hradílek
Stephen Wadeley
Miroslav Svoboda
Eliška Slobodová
David O'Brien

Douglas Silas
Tomáš Čapek
Petr Bokoč
Eva Kopalová
Michael Hideo

Martin Prpič
Petr Kovář
Peter Ondrejka
John Ha
Don Domingo

Red Hat Enterprise Linux 7 System Administrator's Guide

Deployment, Configuration and Administration of Red Hat Enterprise Linux 7

Jaromír Hradílek
Red Hat Engineering Content Services
jhradilek@redhat.com

Douglas Silas
Red Hat Engineering Content Services
silas@redhat.com

Martin Prpič
Red Hat Engineering Content Services
mprpic@redhat.com

Stephen Wadeley
Red Hat Engineering Content Services
swadeley@redhat.com

Tomáš Čapek
Red Hat Engineering Content Services
tcapek@redhat.com

Petr Kovář
Red Hat Engineering Content Services
pkovar@redhat.com

Miroslav Svoboda
Red Hat Engineering Content Services
msvoboda@redhat.com

Petr Bokoc
Red Hat Engineering Content Services
pbokoc@redhat.com

Peter Ondrejka
Red Hat Engineering Content Services
pondrejk@redhat.com

Eliška Sloboďová
Red Hat Engineering Content Services
eslobodo@redhat.com

Eva Kopalová
Red Hat Engineering Content Services

John Ha
Red Hat Engineering Content Services

David O'Brien
Red Hat Engineering Content Services

Michael Hideo
Red Hat Engineering Content Services

Don Domingo
Red Hat Engineering Content Services

Legal Notice

Copyright © 2014 Red Hat, Inc.

This document is licensed by Red Hat under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#). If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, MetaMatrix, Fedora, the Infinity Logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

The System Administrator's Guide documents relevant information regarding the deployment, configuration and administration of Red Hat Enterprise Linux 7. It is oriented towards system administrators with a basic understanding of the system.

Table of Contents

Part I. Basic System Configuration	5
Chapter 1. System Locale and Keyboard Configuration	6
1.1. Setting the System Locale	6
1.2. Changing the Keyboard Layout	7
1.3. Additional Resources	9
Chapter 2. Configuring the Date and Time	11
2.1. Using the timedatectl Command	11
2.2. Using the date Command	14
2.3. Using the hwclock Command	16
2.4. Additional Resources	18
Chapter 3. Managing Users and Groups	20
3.1. Introduction to Users and Groups	20
3.2. Using the User Manager Tool	21
3.3. Using Command Line Tools	25
3.4. Additional Resources	29
Chapter 4. Gaining Privileges	31
4.1. The su Command	31
4.2. The sudo Command	32
4.3. Additional Resources	33
Part II. Package Management	35
Chapter 5. Yum	36
5.1. Checking For and Updating Packages	36
5.2. Working with Packages	39
5.3. Working with Package Groups	48
5.4. Working with Transaction History	51
5.5. Configuring Yum and Yum Repositories	58
5.6. Yum Plug-ins	68
5.7. Additional Resources	71
Part III. Infrastructure Services	72
Chapter 6. Managing Services with systemd	73
6.1. Introduction to systemd	73
6.2. Managing System Services	75
6.3. Working with systemd Targets	81
6.4. Shutting Down, Suspending, and Hibernating the System	86
6.5. Controlling systemd on a Remote Machine	87
6.6. Additional Resources	88
Chapter 7. OpenSSH	90
7.1. The SSH Protocol	90
7.2. Configuring OpenSSH	93
7.3. OpenSSH Clients	100
7.4. More Than a Secure Shell	102
7.5. Additional Resources	104
Chapter 8. TigerVNC	106
8.1. VNC Server	106
8.2. VNC Viewer	108
8.3. Additional Resources	109
Part IV. Servers	110

PART IV. SERVICES	110
Chapter 9. Web Servers	111
9.1. The Apache HTTP Server	111
Chapter 10. Mail Servers	125
10.1. Email Protocols	125
10.2. Email Program Classifications	128
10.3. Mail Transport Agents	129
10.4. Mail Delivery Agents	140
10.5. Mail User Agents	146
10.6. Additional Resources	148
Chapter 11. Directory Servers	150
11.1. OpenLDAP	150
Chapter 12. File and Print Servers	163
12.1. Samba	163
12.2. FTP	174
12.3. Printer Configuration	180
Chapter 13. Configuring NTP Using the chrony Suite	198
13.1. Introduction to the chrony Suite	198
13.2. Understanding chrony and Its Configuration	199
13.3. Using chrony	205
13.4. Setting Up chrony for Different Environments	210
13.5. Using chronyc	211
13.6. Additional Resources	212
Chapter 14. Configuring NTP Using ntpd	213
14.1. Introduction to NTP	213
14.2. NTP Strata	213
14.3. Understanding NTP	214
14.4. Understanding the Drift File	215
14.5. UTC, Timezones, and DST	215
14.6. Authentication Options for NTP	216
14.7. Managing the Time on Virtual Machines	216
14.8. Understanding Leap Seconds	216
14.9. Understanding the ntpd Configuration File	216
14.10. Understanding the ntpd Sysconfig File	218
14.11. Disabling chrony	218
14.12. Checking if the NTP Daemon is Installed	219
14.13. Installing the NTP Daemon (ntpd)	219
14.14. Checking the Status of NTP	219
14.15. Configure the Firewall to Allow Incoming NTP Packets	219
14.16. Configure ntpdate Servers	220
14.17. Configure NTP	221
14.18. Configuring the Hardware Clock Update	226
14.19. Configuring Clock Sources	226
14.20. Additional Resources	227
Chapter 15. Configuring PTP Using ptpt4l	228
15.1. Introduction to PTP	228
15.2. Using PTP	230
15.3. Specifying a Configuration File	232
15.4. Using the PTP Management Client	232
15.5. Synchronizing the Clocks	233
15.6. Verifying Time Synchronization	234
15.7. Serving PTP Time with NTP	236
15.8. Serving NTP Time with PTP	236
15.9. Improving Accuracy	236

15.9. Improving Accuracy	230
15.10. Additional Resources	236
Part V. Monitoring and Automation	238
Chapter 16. System Monitoring Tools	239
16.1. Viewing System Processes	239
16.2. Viewing Memory Usage	242
16.3. Viewing CPU Usage	244
16.4. Viewing Block Devices and File Systems	244
16.5. Viewing Hardware Information	249
16.6. Monitoring Performance with Net-SNMP	252
16.7. Additional Resources	262
Chapter 17. OpenLMI	263
17.1. About OpenLMI	263
17.2. Installing OpenLMI	264
17.3. Configuring SSL Certificates for OpenPegasus	265
17.4. Using LMIShell	270
17.5. Using OpenLMI Scripts	308
17.6. Additional Resources	309
Chapter 18. Viewing and Managing Log Files	310
18.1. Locating Log Files	310
18.2. Basic Configuration of Rsyslog	310
18.3. Working with Queues in Rsyslog	326
18.4. Using Rsyslog Modules	331
18.5. Interaction of Rsyslog and Journal	334
18.6. Structured Logging with Rsyslog	335
18.7. Debugging Rsyslog	338
18.8. Using the Journal	338
18.9. Managing Log Files in Graphical Environment	344
18.10. Additional Resources	347
Chapter 19. Automating System Tasks	349
19.1. Cron and Anacron	349
19.2. At and Batch	354
19.3. Additional Resources	358
Chapter 20. Automatic Bug Reporting Tool (ABRT)	359
20.1. Introduction to ABRT	359
20.2. Installing ABRT and Starting its Services	359
20.3. Configuring ABRT	361
20.4. Detecting Software Problems	366
20.5. Handling Detected Problems	368
20.6. Additional Resources	370
Chapter 21. OProfile	372
21.1. Overview of Tools	372
21.2. Using perf	373
21.3. Configuring OProfile Using Legacy Mode	376
21.4. Starting and Stopping OProfile Using Legacy Mode	381
21.5. Saving Data in Legacy Mode	381
21.6. Analyzing the Data	382
21.7. Understanding /dev/oprofile/	386
21.8. Example Usage	387
21.9. OProfile Support for Java	387
21.10. Graphical Interface	388
21.11. OProfile and SystemTap	391
21.12. Additional Resources	391

Part VI. Kernel, Module and Driver Configuration	392
Chapter 22. Working with the GRUB 2 Boot Loader	393
22.1. Configuring the GRUB 2 Boot Loader	393
22.2. Customizing GRUB 2 Menu	394
22.3. GRUB 2 Password Protection	398
22.4. Reinstalling GRUB 2	400
22.5. GRUB 2 over Serial Console	401
22.6. Terminal Menu Editing During Boot	402
22.7. UEFI Secure Boot	403
22.8. Additional Resources	404
Chapter 23. Manually Upgrading the Kernel	405
23.1. Overview of Kernel Packages	405
23.2. Preparing to Upgrade	406
23.3. Downloading the Upgraded Kernel	407
23.4. Performing the Upgrade	407
23.5. Verifying the Initial RAM Disk Image	408
23.6. Verifying the Boot Loader	410
Chapter 24. Working with Kernel Modules	411
24.1. Listing Currently-Loaded Modules	411
24.2. Displaying Information About a Module	412
24.3. Loading a Module	414
24.4. Unloading a Module	415
24.5. Setting Module Parameters	416
24.6. Persistent Module Loading	417
24.7. Signing Kernel Modules for Secure Boot	418
24.8. Additional Resources	424
RPM	425
A.1. RPM Design Goals	425
A.2. Using RPM	426
A.3. Finding and Verifying RPM Packages	431
A.4. Practical and Common Examples of RPM Usage	433
A.5. Additional Resources	433
The X Window System	435
B.1. The X Server	435
B.2. Desktop Environments and Window Managers	435
B.3. X Server Configuration Files	437
B.4. Fonts	445
B.5. Runlevels, targets, and X	446
B.6. Additional Resources	446
Revision History	447
C.1. Acknowledgments	447
Index	447

Part I. Basic System Configuration

This part covers basic system administration tasks such as keyboard configuration, date and time configuration, managing users and groups, and gaining privileges.

Chapter 1. System Locale and Keyboard Configuration

The *system locale* specifies the language settings of system services and user interfaces. The *keyboard layout* settings control the layout used on the text console and graphical user interfaces.

These settings can be made by modifying the `/etc/locale.conf` configuration file or by using the `localectl` utility. Also, you can use the graphical user interface to perform the task; for a description of this method, see [Red Hat Enterprise Linux 7 Installation Guide](#).

1.1. Setting the System Locale

System-wide locale settings are stored in the `/etc/locale.conf` file, which is read at early boot by the `systemd` daemon. The locale settings configured in `/etc/locale.conf` are inherited by every service or user, unless individual programs or individual users override them.

The basic file format of `/etc/locale.conf` is a newline-separated list of variable assignments. For example, German locale with English messages in `/etc/locale.conf` looks as follows:

```
LANG=de_DE.UTF-8
LC_MESSAGES=C
```

Here, the `LC_MESSAGES` option determines the locale used for diagnostic messages written to the standard error output. To further specify locale settings in `/etc/locale.conf`, you can use several other options, most relevant are summarized in [Table 1.1, “Options configurable in /etc/locale.conf”](#). See the `locale(7)` manual page for detailed information on these options. Note that the `LC_ALL` option, which represents all possible options, should not be configured in `/etc/locale.conf`.

Table 1.1. Options configurable in /etc/locale.conf

Option	Description
<code>LANG</code>	Provides a default value for the system locale.
<code>LC_COLLATE</code>	Changes the behavior of functions which compare strings in the local alphabet.
<code>LC_CTYPE</code>	Changes the behavior of the character handling and classification functions and the multibyte character functions.
<code>LC_NUMERIC</code>	Describes the way numbers are usually printed, with details such as decimal point versus decimal comma.
<code>LC_TIME</code>	Changes the display of the current time, 24-hour versus 12-hour clock.
<code>LC_MESSAGES</code>	Determines the locale used for diagnostic messages written to the standard error output.

1.1.1. Displaying the Current Status

The `localectl` command can be used to query and change the system locale and keyboard layout settings. To show the current settings, use the `status` option:

```
localectl status
```

Example 1.1. Displaying the Current Status

The output of the previous command lists the currently set locale, keyboard layout configured for the console and for the X11 window system.

```
~]$ localectl status
System Locale: LANG=en_US.UTF-8
    VC Keymap: us
    X11 Layout: n/a
```

1.1.2. Listing Available Locales

To list all locales available for your system, type:

```
localectl list-locales
```

Example 1.2. Listing Locales

Imagine you want to select a specific English locale, but you are not sure if it is available on the system. You can check that by listing all English locales with the following command:

```
~]$ localectl list-locales | grep en_
en_AG
en_AG.utf8
en_AU
en_AU.iso88591
en_AU.utf8
en_BW
en_BW.iso88591
en_BW.utf8

output truncated
```

1.1.3. Setting the Locale

To set the default system locale, use the following command as a **root**:

```
localectl set-locale LANG=locale
```

Replace *locale* with the locale name, found with **list-locales**. With this command, you can also set options from [Table 1.1, “Options configurable in /etc/locale.conf”](#).

Example 1.3. Changing the Default Locale

For example, if you want to set British English as your default locale, first find the name of this locale by using **list-locales**. Then, as a **root** user, type the command in the following form:

```
~]# localectl set-locale LANG=en_GB.utf8
```

1.2. Changing the Keyboard Layout

Keyboard Layout Configuration

The keyboard layout settings let the user to control the layout used on the text console and graphical user interfaces.

1.2.1. Displaying the Current Settings

As mentioned before, you can check your current keyboard layout configuration with the following command:

```
localectl status
```

Example 1.4. Displaying the Keyboard Settings

In the following output, you can see the keyboard layout configured for the virtual console and for the X11 window system.

```
~]$ localectl status
System Locale: LANG=en_US.utf8
VC Keymap: us
X11 Layout: us
```

1.2.2. Listing Available Keypmaps

To list all available keyboard layouts that can be configured on your system, type:

```
localectl list-keymaps
```

Example 1.5. Searching for a Particular Keypad

You can use **grep** to search the output of the previous command for a specific keymap name. There are often multiple keymaps compatible with your currently set locale. For example, to find available Czech keyboard layouts, type:

```
~]$ localectl list-keymaps | grep cz
cz
cz-cp1250
cz-lat2
cz-lat2-prog
cz-qwerty
cz-us-qwertz
sunt5-cz-us
sunt5-us-cz
```

1.2.3. Setting the Keypad

To set the default keyboard layout for your system, use the following command as a **root**:

```
localectl set-keymap map
```

Replace *map* with the name of keymap taken from the output of **list-keymaps**. Unless the **--no-convert** option is passed, the selected setting is also applied to the default keyboard mapping of the X11 window system, after converting it to the closest matching X11 keyboard mapping. This also applies reversely, you can specify the both keymaps with the following command (as a **root**):

```
localectl set-x11-keymap map
```

If you want your X11 layout to differ from the console layout, use the **--no-convert** option

```
localectl --no-convert set-x11-keymap map
```

With this option, the X11 keymap is specified without changing the previous console layout setting.

Example 1.6. Setting the X11 Keymap Separately

Imagine you want to use German keyboard layout in the graphical interface, but for console operations you want to retain the US keymap. To do so, type (as a **root**):

```
~]# localectl --no-convert set-x11-keymap de
```

Then you can verify if your setting was successful by checking the current status:

```
~]$ localectl status
System Locale: LANG=de_DE.UTF-8
    VC Keymap: us
    X11 Layout: de
```

Apart from keyboard layout (*map*), three other options can be specified:

```
localectl set-x11-keymap map model variant options
```

Replace *model* with the keyboard model name, *variant* and *options* with keyboard variant and option components, which can be used to enhance the keyboard behavior. These options are not set by default. For more information on X11 Model, X11 Variant and X11 Options see **kbd(4)** man page.

1.3. Additional Resources

For more information on how to configure the keyboard layout on Red Hat Enterprise Linux, see the resources listed below.

Installed Documentation

- ▶ **localectl(1)** — The manual page for the **localectl** command line utility documents how to use this tool to configure the system locale and keyboard layout.
- ▶ **loadkeys(1)** — The manual page for the **loadkeys** command provides more information on how to use this tool to change the keyboard layout in a virtual console.

See Also

- ▶ [Chapter 4, Gaining Privileges](#) documents how to gain administrative privileges by using the **su** and **sudo** commands.

- ▶ [Chapter 6, *Managing Services with systemd*](#) provides more information on systemd and documents how to use the **systemctl** command to manage system services.

Chapter 2. Configuring the Date and Time

Modern operating systems distinguish between the following two types of clocks:

- ▶ A *real-time clock* (RTC), commonly referred to as a *hardware clock*, (typically an integrated circuit on the system board) that is completely independent of the current state of the operating system and runs even when the computer is shut down.
- ▶ A *system clock*, also known as a *software clock*, that is maintained by the kernel and its initial value is based on the real-time clock. Once the system is booted and the system clock is initialized, the system clock is completely independent of the real-time clock.

The real-time clock can use either *local time* or *Coordinated Universal Time* (UTC). If you configure the real-time clock to use UTC, the system time is calculated by applying the offset for your time zone and if applicable, also *daylight saving time* (DST). In comparison, local time represents the actual time in your current time zone. In most cases, it is recommended that you use UTC.

Red Hat Enterprise Linux 7 offers three command line tools that can be used to configure and display information about the system date and time: the **timedatectl** utility, which is new in Red Hat Enterprise Linux 7 and is part of **systemd**; the traditional **date** command; and the **hwclock** utility for accessing the hardware clock.

2.1. Using the **timedatectl** Command

The **timedatectl** utility is distributed as part of the **systemd** system and service manager and allows you to review and change the configuration of the system clock. You can use this tool to change the current date and time, set the time zone, or enable automatic synchronization of the system clock with a remote server.

For information on how to display the current date and time in a custom format, see also [Section 2.2, “Using the **date** Command”](#).

2.1.1. Displaying the Current Date and Time

To display the current date and time along with detailed information about the configuration of the system and hardware clock, run the **timedatectl** command with no additional command line options:

```
timedatectl
```

This displays the local, universal, and RTC time, the currently used time zone, the status of the Network Time Protocol (**NTP**) configuration, and additional information related to DST.

Example 2.1. Displaying the Current Date and Time

The following is an example output of the **timedatectl** command on a system that does not use **NTP** to synchronize the system clock with a remote server:

```
~]$ timedatectl
    Local time: Mon 2013-09-16 19:30:24 CEST
    Universal time: Mon 2013-09-16 17:30:24 UTC
          Timezone: Europe/Prague (CEST, +0200)
        NTP enabled: no
    NTP synchronized: no
      RTC in local TZ: no
        DST active: yes
Last DST change: DST began at
                  Sun 2013-03-31 01:59:59 CET
                  Sun 2013-03-31 03:00:00 CEST
Next DST change: DST ends (the clock jumps one hour backwards) at
                  Sun 2013-10-27 02:59:59 CEST
                  Sun 2013-10-27 02:00:00 CET
```

2.1.2. Changing the Current Date

To change the current date, type the following at a shell prompt as **root**:

```
timedatectl set-time YYYY-MM-DD
```

Replace *YYYY* with a four-digit year, *MM* with a two-digit month, and *DD* with a two-digit day of the month.

Example 2.2. Changing the Current Date

To change the current date to 2 June 2013, run the following command as **root**:

```
~]# timedatectl set-time 2013-06-02
```

2.1.3. Changing the Current Time

To change the current time, type the following at a shell prompt as **root**:

```
timedatectl set-time HH:MM:SS
```

Replace *HH* with an hour, *MM* with a minute, and *SS* with a second, all typed in a two-digit form.

By default, the system is configured to use UTC. To configure your system to maintain the clock in the local time, run the **timedatectl** command with the **set-local-rtc** option as **root**:

```
timedatectl set-local-rtc boolean
```

To configure your system to maintain the clock in the local time, replace *boolean* with **yes**. To configure the system to use UTC, replace *boolean* with **no** (the default option).

Example 2.3. Changing the Current Time

To change the current time to 11:26 p.m., run the following command as **root**:

```
~]# timedatectl set-time 23:26:00
```

2.1.4. Changing the Time Zone

To list all available time zones, type the following at a shell prompt:

```
timedatectl list-timezones
```

To change the currently used time zone, type as **root**:

```
timedatectl set-timezone time_zone
```

Replace *time_zone* with any of the values listed by the **timedatectl list-timezones** command.

Example 2.4. Changing the Time Zone

To identify which time zone is closest to your present location, use the **timedatectl** command with the **list-timezones** command line option. For example, to list all available time zones in Europe, type:

```
~]# timedatectl list-timezones | grep Europe
Europe/Amsterdam
Europe/Andorra
Europe/Athens
Europe/Belgrade
Europe/Berlin
Europe/Bratislava
...
```

To change the time zone to **Europe/Prague**, type as **root**:

```
~]# timedatectl set-timezone Europe/Prague
```

2.1.5. Synchronizing the System Clock with a Remote Server

As opposed to the manual setup described in the previous sections, the **timedatectl** command also allows you to enable automatic synchronization of your system clock with a remote server using **NTP**. To enable or disable this feature, type the following at a shell prompt as **root**:

```
timedatectl set-ntp boolean
```

To configure your system to synchronize the system clock with a remote **NTP** server, replace *boolean* with **yes** (the default option). To disable this feature, replace *boolean* with **no**.

Example 2.5. Synchronizing the System Clock with a Remote Server

To enable automatic synchronization of the system clock with a remote server, type:

```
~]# timedatectl set-ntp yes
```

2.2. Using the date Command

The **date** utility is available on all Linux systems and allows you to display and configure the current date and time. It is frequently used in scripts to display detailed information about the system clock in a custom format.

For information on how to change the time zone or enable automatic synchronization of the system clock with a remote server, see [Section 2.1, “Using the timedatectl Command”](#).

2.2.1. Displaying the Current Date and Time

To display the current date and time, run the **date** command with no additional command line options:

```
date
```

This displays the day of the week followed by the current date, local time, abbreviated time zone, and year.

By default, the **date** command displays the local time. To display the time in UTC, run the command with the **--utc** or **-u** command line option:

```
date --utc
```

You can also customize the format of the displayed information by providing the **+“format”** option on the command line:

```
date +"format"
```

Replace *format* with one or more supported control sequences as illustrated in [Example 2.6, “Displaying the Current Date and Time”](#). See [Table 2.1, “Commonly Used Control Sequences”](#) for a list of the most frequently used formatting options, or the **date(1)** manual page for a complete list of these options.

Table 2.1. Commonly Used Control Sequences

Control Sequence	Description
%H	The hour in the <i>HH</i> format (for example, 17).
%M	The minute in the <i>MM</i> format (for example, 30).
%S	The second in the <i>SS</i> format (for example, 24).
%d	The day of the month in the <i>DD</i> format (for example, 16).
%m	The month in the <i>MM</i> format (for example, 09).
%Y	The year in the <i>YYYY</i> format (for example, 2013).
%Z	The time zone abbreviation (for example, CEST).
%F	The full date in the <i>YYYY-MM-DD</i> format (for example, 2013-09-16). This option is equal to %Y-%m-%d .

Control Sequence	Description
%T	The full time in the <i>HH:MM:SS</i> format (for example, 17:30:24). This option is equal to %H:%M:%S

Example 2.6. Displaying the Current Date and Time

To display the current date and time in UTC, type the following at a shell prompt:

```
~]$ date --utc
Mon Sep 16 17:30:24 CEST 2013
```

To customize the output of the **date** command, type:

```
~]$ date +"%Y-%m-%d %H:%M"
2013-09-16 17:30
```

2.2.2. Changing the Current Date

To change the current date, type the following at a shell prompt as **root**:

```
date +%F -s YYYY-MM-DD
```

Replace *YYYY* with a four-digit year, *MM* with a two-digit month, and *DD* with a two-digit day of the month.

Example 2.7. Changing the Current Date

To change the current date to 2 June 2013, run the following command as **root**:

```
~]# date +%F -s 2013-06-02
```

2.2.3. Changing the Current Time

To change the current time, run the **date** command with the **--set** or **-s** option as **root**:

```
date +%T -s HH:MM:SS
```

Replace *HH* with an hour, *MM* with a minute, and *SS* with a second, all typed in a two-digit form.

By default, the **date** command sets the system clock in the local time. To set the system clock in UTC instead, run the command with the **--utc** or **-u** command line option:

```
date +%T --set HH:MM:SS --utc
```

Example 2.8. Changing the Current Time

To change the current time to 11:26 p.m., run the following command as **root**:

```
~]# date +%T --set 23:26:00
```

2.3. Using the **hwclock** Command

hwclock is a utility for accessing the hardware clock. Hardware clock is independent of the operation system you use and works even when the machine is shut down. This program is used to find out the time from the hardware clock and set the system time at boot time. **hwclock** also contains facilities for compensating for systematic drift in the hardware clock.

hwclock stores the values of: year, month, day, hour, minute, and seconds. It is not able to store the time standard, localtime or Coordinated Universal Time (UTC), nor set the Daylight Saving Time (DST).



Note

In Red Hat Enterprise Linux 6, the **hwclock** command was run automatically on every system shutdown or reboot, but it is not in Red Hat Enterprise Linux 7.

2.3.1. Displaying the Current Date and Time

Running **hwclock** with no command line options as **root** returns a string of current date and time information:

```
hwclock
```

Example 2.9. Displaying the Current Date and Time

The **hwclock** command returns the time on standard output. Even if you keep your hardware clock in UTC, the displayed time is by default the local time. You can check the default setting by opening the **/etc/adjtime** file.

```
~]# hwclock
Tue 15 Apr 2014 04:23:46 PM CEST -0.329272 seconds
```

CEST is a time zone abbreviation and stands for Central European Summer Time.

In order to get UTC string with date and time information of the machine, execute:

```
hwclock --utc
```

Example 2.10. Using the --utc Option

The output of the **hwclock --utc** command shows exactly the same as in [Example 2.9, “Displaying the Current Date and Time”](#) as it displays the current UTC specification.

```
~]# hwclock --utc
Tue 15 Apr 2014 04:23:46 PM CEST -0.329272 seconds
```

To display the local date and time, make use of the **--localtime** option.

```
hwclock --localtime
```

Example 2.11. Using the --localtime option

To display the local date and time type the **hwclock --localtime** command:

```
~]# hwclock --localtime
Tue 15 Apr 2014 02:23:46 PM CEST -0.329272 seconds
```

 **Note**

The decision whether to keep your clock in UTC or local time is up to you. Just remember that if you run **hwclock --localtime** and your hardware clock is set to UTC, **hwclock** returns a time shift comprising the difference between UTC and local time. The original setting or RTC time will not change.

2.3.2. Setting and Synchronizing the Date and Time

Besides viewing the date and time, you can manually set the **hwclock** utility to a specific time or to synchronize the system time to hardware clock or vice versa.

When you need to change the hardware clock date and time, you can do so by appending **--set** and **--date** options along with your specification:

```
hwclock --set --date "dd mmm yyyy HH:MM"
```

Replace *dd* with a day (a two-digit number), *mmm* with a month (a three-letter abbreviation), *yyyy* with a year (a four-digit number), *HH* with an hour (a two-digit number), *MM* with a minute (a two-digit number).

Example 2.12. Setting the Hardware Clock to a Specific Date and Time

If you wish to set the date and time to a specific value, for example, to "21:17, October 21, 2014", run the command in the following format:

```
~]# hwclock --set --date "21 Oct 2014 21:17"
```

You can also set the hardware clock to the current system time making use of synchronization. The following command synchronizes the system time to hardware clock:

```
hwclock --systohc
```

There are two directions of readjustment:

- ▶ You can readjust the hardware clock to the system time to point to the local time:

```
hwclock --systohc --localtime
```

- ▶ Or, you can readjust the hardware clock to the system time to point to UTC:

```
hwclock --systohc --utc
```

This command updates the **rtc** utility, which saves the new setting and displays it when you run **hwclock --localtime** or **hwclock --utc** respectively. The same function is provided by the **hwclock --adjust** command.

This setting is particularly useful if you are booting multiple Linux distributions on the same machine and you face the problem of inconsistent time zone settings across the distributions.

Example 2.13. Synchronizing System Time to Hardware Clock

To synchronize the system time to hardware clock, run the following command as **root**:

```
~]# hwclock --systohc --localtime
```

For the changes to take effect, reboot your system.

In addition, you can carry out the inverse operation, to set the system time from the hardware clock, by executing:

```
hwclock --hctosys
```

Example 2.14. Synchronizing Hardware Clock to System Time

To synchronize hardware clock to system time, run the following command as **root**:

```
~]# hwclock --hctosys --localtime
```

For the changes to take effect, reboot your system.

2.4. Additional Resources

For more information on how to configure the date and time in Red Hat Enterprise Linux 7, see the resources listed below.

Installed Documentation

- ▶ **timedatectl(1)** — The manual page for the **timedatectl** command line utility documents how to use this tool to query and change the system clock and its settings.
- ▶ **date(1)** — The manual page for the **date** command provides a complete list of supported command line options.
- ▶ **hwclock(8)** — The manual page for the **hwclock** command provides a complete list of supported command line options.

See Also

- ▶ [Chapter 1, System Locale and Keyboard Configuration](#) documents how to configure the keyboard layout.
- ▶ [Chapter 4, Gaining Privileges](#) documents how to gain administrative privileges by using the **su** and **sudo** commands.

- ▶ [Chapter 6, Managing Services with `systemd`](#) provides more information on `systemd` and documents how to use the `systemctl` command to manage system services.

Chapter 3. Managing Users and Groups

The control of users and groups is a core element of Red Hat Enterprise Linux system administration. This chapter explains how to add, manage, and delete users and groups in the graphical user interface and on the command line, and covers advanced topics, such as enabling password aging or creating group directories.

3.1. Introduction to Users and Groups

While users can be either people (meaning accounts tied to physical users) or accounts which exist for specific applications to use, groups are logical expressions of organization, tying users together for a common purpose. Users within a group can read, write, or execute files owned by that group.

Each user is associated with a unique numerical identification number called a *user ID* (UID). Likewise, each group is associated with a *group ID* (GID). A user who creates a file is also the owner and group owner of that file. The file is assigned separate read, write, and execute permissions for the owner, the group, and everyone else. The file owner can be changed only by **root**, and access permissions can be changed by both the **root** user and file owner.

Additionally, Red Hat Enterprise Linux supports *access control lists* (ACLs) for files and directories which allow permissions for specific users outside of the owner to be set. For more information about this feature, see the [Access Control Lists](#) chapter of the [Storage Administration Guide](#).

3.1.1. User Private Groups

Red Hat Enterprise Linux uses a *user private group* (UPG) scheme, which makes UNIX groups easier to manage. A user private group is created whenever a new user is added to the system. It has the same name as the user for which it was created and that user is the only member of the user private group.

User private groups make it safe to set default permissions for a newly created file or directory, allowing both the user and *the group of that user* to make modifications to the file or directory.

The setting which determines what permissions are applied to a newly created file or directory is called a *umask* and is configured in the **/etc/bashrc** file. Traditionally on UNIX systems, the **umask** is set to **022**, which allows only the user who created the file or directory to make modifications. Under this scheme, all other users, *including members of the creator's group*, are not allowed to make any modifications. However, under the UPG scheme, this “group protection” is not necessary since every user has their own private group.

3.1.2. Shadow Passwords

In environments with multiple users, it is very important to use *shadow passwords* provided by the *shadow-utils* package to enhance the security of system authentication files. For this reason, the installation program enables shadow passwords by default.

The following is a list of the advantages shadow passwords have over the traditional way of storing passwords on UNIX-based systems:

- ▶ Shadow passwords improve system security by moving encrypted password hashes from the world-readable **/etc/passwd** file to **/etc/shadow**, which is readable only by the **root** user.
- ▶ Shadow passwords store information about password aging.
- ▶ Shadow passwords allow the **/etc/login.defs** file to enforce security policies.

Most utilities provided by the `shadow-utils` package work properly whether or not shadow passwords are enabled. However, since password aging information is stored exclusively in the `/etc/shadow` file, any commands which create or modify password aging information do not work. The following is a list of utilities and commands that do not work without first enabling shadow passwords:

- ▶ The `chage` utility.
- ▶ The `gpasswd` utility.
- ▶ The `usermod` command with the `-e` or `-f` option.
- ▶ The `useradd` command with the `-e` or `-f` option.

3.2. Using the User Manager Tool

The **User Manager** application allows you to view, modify, add, and delete local users and groups in the graphical user interface. To start the application, either select **System → Administration → Users and Groups** from the panel, or type `system-config-users` at a shell prompt. Note that unless you have superuser privileges, the application will prompt you to authenticate as `root`.

3.2.1. Viewing Users and Groups

The main window of the **User Manager** is divided into two tabs: The **Users** tab provides a list of local users along with additional information about their user ID, primary group, home directory, login shell, and full name. The **Groups** tab provides a list of local groups with information about their group ID and group members.

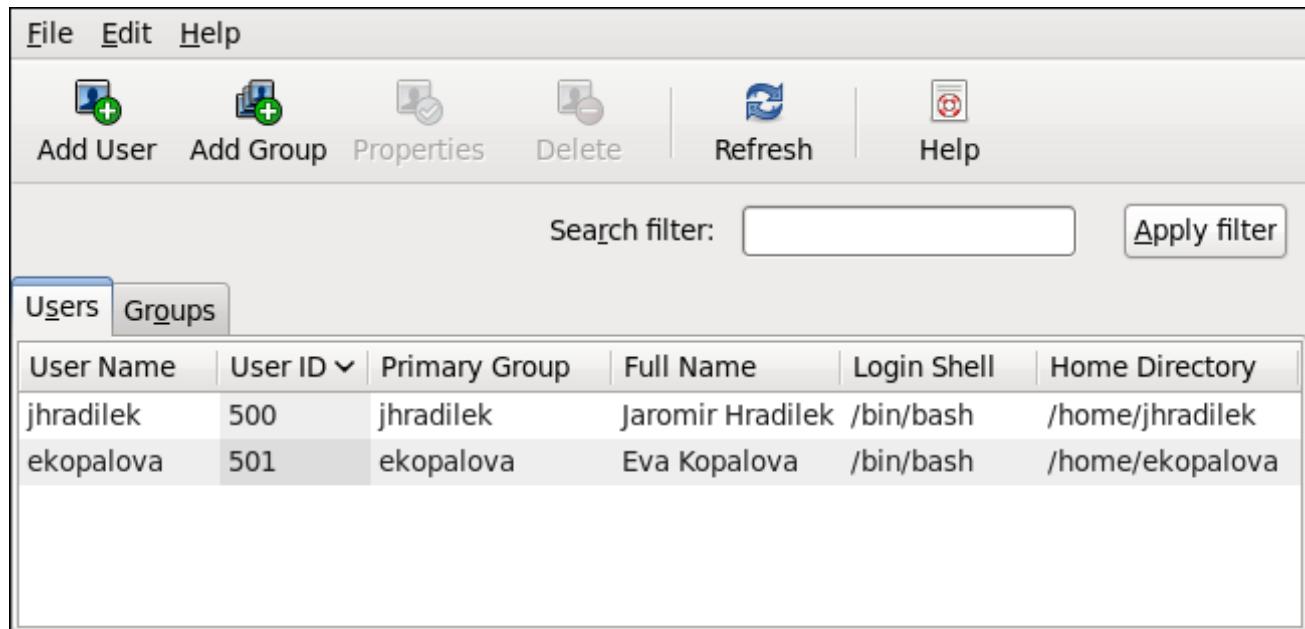


Figure 3.1. Viewing users and groups

To find a specific user or group, type the first few letters of the name in the **Search filter** field and either press **Enter**, or click the **Apply filter** button. You can also sort the items according to any of the available columns by clicking the column header.

Red Hat Enterprise Linux reserves user and group IDs below 500 for system users and groups. By default, the **User Manager** does not display the system users. To view all users and groups, select **Edit** → **Preferences** to open the **Preferences** dialog box, and clear the **Hide system users and groups** checkbox.

3.2.2. Adding a New User

To add a new user, click the **Add User** button. A window as shown in [Figure 3.2, “Adding a new user”](#) appears.

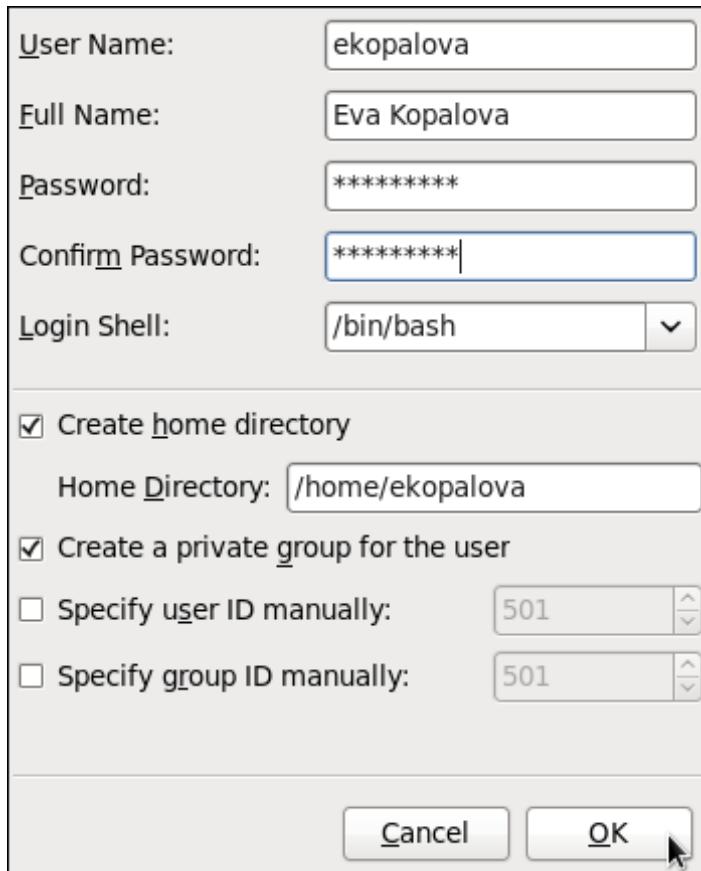


Figure 3.2. Adding a new user

The **Add New User** dialog box allows you to provide information about the newly created user. In order to create a user, enter the username and full name in the appropriate fields and then type the user's password in the **Password** and **Confirm Password** fields. The password must be at least six characters long.

Password security advice

It is advisable to use a much longer password, as this makes it more difficult for an intruder to guess it and access the account without permission. It is also recommended that the password not be based on a dictionary term: use a combination of letters, numbers and special characters.

The **Login Shell** pulldown list allows you to select a login shell for the user. If you are not sure which shell to select, accept the default value of **/bin/bash**.

By default, the **User Manager** application creates the home directory for a new user in `/home/username/`. You can choose not to create the home directory by clearing the **Create home directory** checkbox, or change this directory by editing the content of the **Home Directory** text box. Note that when the home directory is created, default configuration files are copied into it from the `/etc/skel/` directory.

Red Hat Enterprise Linux uses a user private group (UPG) scheme. Whenever you create a new user, a unique group with the same name as the user is created by default. If you do not want to create this group, clear the **Create a private group for the user** checkbox.

To specify a user ID for the user, select **Specify user ID manually**. If the option is not selected, the next available user ID above 500 is assigned to the new user. Because Red Hat Enterprise Linux reserves user IDs below 500 for system users, it is not advisable to manually assign user IDs 1–499.

Clicking the **OK** button creates the new user. To configure more advanced user properties, such as password expiration, modify the user's properties after adding the user.

3.2.3. Adding a New Group

To add a new user group, select **Add Group** from the toolbar. A window similar to [Figure 3.3, “New Group”](#) appears. Type the name of the new group. To specify a group ID for the new group, select **Specify group ID manually** and select the GID. Note that Red Hat Enterprise Linux also reserves group IDs lower than 500 for system groups.

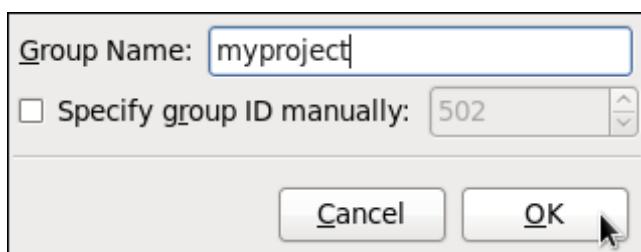


Figure 3.3. New Group

Click **OK** to create the group. The new group appears in the group list.

3.2.4. Modifying User Properties

To view the properties of an existing user, click on the **Users** tab, select the user from the user list, and click **Properties** from the menu (or choose **File → Properties** from the pulldown menu). A window similar to [Figure 3.4, “User Properties”](#) appears.

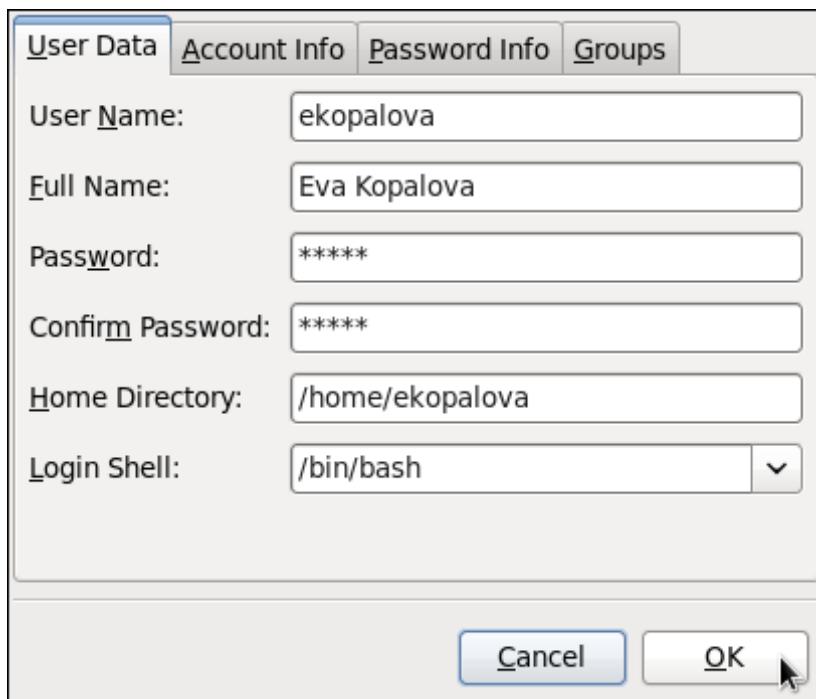


Figure 3.4. User Properties

The **User Properties** window is divided into multiple tabbed pages:

- ▶ **User Data** — Shows the basic user information configured when you added the user. Use this tab to change the user's full name, password, home directory, or login shell.
- ▶ **Account Info** — Select **Enable account expiration** if you want the account to expire on a certain date. Enter the date in the provided fields. Select **Local password is locked** to lock the user account and prevent the user from logging into the system.
- ▶ **Password Info** — Displays the date that the user's password last changed. To force the user to change passwords after a certain number of days, select **Enable password expiration** and enter a desired value in the **Days before change required:** field. The number of days before the user's password expires, the number of days before the user is warned to change passwords, and days before the account becomes inactive can also be changed.
- ▶ **Groups** — Allows you to view and configure the Primary Group of the user, as well as other groups that you want the user to be a member of.

3.2.5. Modifying Group Properties

To view the properties of an existing group, select the group from the group list and click **Properties** from the menu (or choose **File** → **Properties** from the pulldown menu). A window similar to [Figure 3.5, “Group Properties”](#) appears.

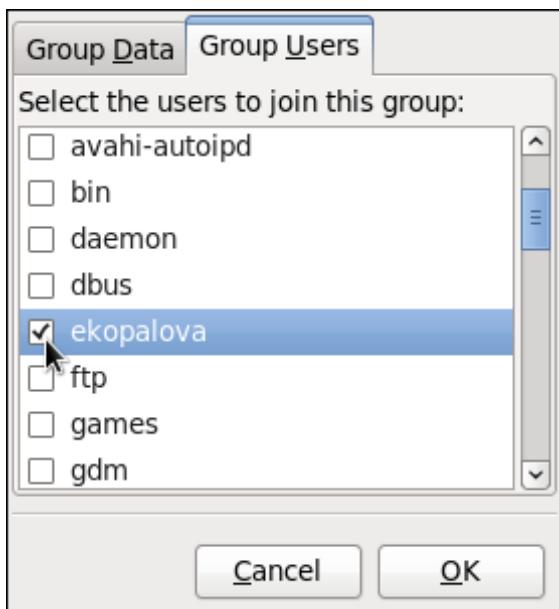


Figure 3.5. Group Properties

The **Group Users** tab displays which users are members of the group. Use this tab to add or remove users from the group. Click **OK** to save your changes.

3.3. Using Command Line Tools

The easiest way to manage users and groups on Red Hat Enterprise Linux is to use the **User Manager** application as described in [Section 3.2, “Using the User Manager Tool”](#). However, if you prefer command line tools or do not have the X Window System installed, you can use command line utilities that are listed in [Table 3.1, “Command line utilities for managing users and groups”](#).

Table 3.1. Command line utilities for managing users and groups

Utilities	Description
useradd, usermod, userdel	Standard utilities for adding, modifying, and deleting user accounts.
groupadd, groupmod, groupdel	Standard utilities for adding, modifying, and deleting groups.
gpasswd	Standard utility for administering the /etc/group configuration file.
pwck, grpck	Utilities that can be used for verification of the password, group, and associated shadow files.
pwconv, pwunconv	Utilities that can be used for the conversion of passwords to shadow passwords, or back from shadow passwords to standard passwords.

3.3.1. Adding a New User

To add a new user to the system, typing the following at a shell prompt as **root**:

```
useradd [options] username
```

...where *options* are command line options as described in [Table 3.2, “useradd command line options”](#).

By default, the **useradd** command creates a locked user account. To unlock the account, run the following

command as **root** to assign a password:

```
passwd username
```

Optionally, you can set password aging policy. See *Red Hat Enterprise Linux 7 Security Guide* for information on how to enable password aging.

Table 3.2. useradd command line options

Option	Description
-c ' <i>comment</i> '	<i>comment</i> can be replaced with any string. This option is generally used to specify the full name of a user.
-d <i>home_directory</i>	Home directory to be used instead of default /home/username/ .
-e <i>date</i>	Date for the account to be disabled in the format YYYY-MM-DD.
-f <i>days</i>	Number of days after the password expires until the account is disabled. If 0 is specified, the account is disabled immediately after the password expires. If -1 is specified, the account is not be disabled after the password expires.
-g <i>group_name</i>	Group name or group number for the user's default group. The group must exist prior to being specified here.
-G <i>group_list</i>	List of additional (other than default) group names or group numbers, separated by commas, of which the user is a member. The groups must exist prior to being specified here.
-m	Create the home directory if it does not exist.
-M	Do not create the home directory.
-N	Do not create a user private group for the user.
-p <i>password</i>	The password encrypted with crypt .
-r	Create a system account with a UID less than 500 and without a home directory.
-s	User's login shell, which defaults to /bin/bash .
-u <i>uid</i>	User ID for the user, which must be unique and greater than 499.

Explaining the Process

The following steps illustrate what happens if the command **useradd juan** is issued on a system that has shadow passwords enabled:

1. A new line for **juan** is created in **/etc/passwd**:

```
juan:x:501:501::/home/juan:/bin/bash
```

The line has the following characteristics:

- ▶ It begins with the username **juan**.
- ▶ There is an **x** for the password field indicating that the system is using shadow passwords.
- ▶ A UID greater than 499 is created. Under Red Hat Enterprise Linux, UIDs below 500 are reserved for system use and should not be assigned to users.
- ▶ A GID greater than 499 is created. Under Red Hat Enterprise Linux, GIDs below 500 are reserved for system use and should not be assigned to users.

- ▶ The optional GECOS information is left blank. The GECOS field can be used to provide additional information about the user, such as their full name or phone number.
- ▶ The home directory for **juan** is set to **/home/juan/**.
- ▶ The default shell is set to **/bin/bash**.

2. A new line for **juan** is created in **/etc/shadow**:

```
juan:!!:14798:0:99999:7:::
```

The line has the following characteristics:

- ▶ It begins with the username **juan**.
- ▶ Two exclamation marks (! !) appear in the password field of the **/etc/shadow** file, which locks the account.



Note

If an encrypted password is passed using the **-p** flag, it is placed in the **/etc/shadow** file on the new line for the user.

- ▶ The password is set to never expire.

3. A new line for a group named **juan** is created in **/etc/group**:

```
juan:x:501:
```

A group with the same name as a user is called a *user private group*. For more information on user private groups, see [Section 3.1.1, “User Private Groups”](#).

The line created in **/etc/group** has the following characteristics:

- ▶ It begins with the group name **juan**.
- ▶ An **x** appears in the password field indicating that the system is using shadow group passwords.
- ▶ The GID matches the one listed for user **juan** in **/etc/passwd**.

4. A new line for a group named **juan** is created in **/etc/gshadow**:

```
juan:!::
```

The line has the following characteristics:

- ▶ It begins with the group name **juan**.
- ▶ An exclamation mark (!) appears in the password field of the **/etc/gshadow** file, which locks the group.
- ▶ All other fields are blank.

5. A directory for user **juan** is created in the **/home/** directory:

```
~]# ls -l /home
total 4
drwx----- 4 juan juan 4096 Mar  3 18:23 juan
```

This directory is owned by user **juan** and group **juan**. It has *read*, *write*, and *execute* privileges *only* for the user **juan**. All other permissions are denied.

- The files within the **/etc/skel/** directory (which contain default user settings) are copied into the new **/home/juan/** directory:

```
~]# ls -la /home/juan
total 28
drwx----- 4 juan juan 4096 Mar  3 18:23 .
drwxr-xr-x  5 root root 4096 Mar  3 18:23 ..
-rw-r--r--  1 juan juan   18 Jun 22  2010 .bash_logout
-rw-r--r--  1 juan juan  176 Jun 22  2010 .bash_profile
-rw-r--r--  1 juan juan  124 Jun 22  2010 .bashrc
drwxr-xr-x  2 juan juan 4096 Jul 14  2010 .gnome2
drwxr-xr-x  4 juan juan 4096 Nov 23 15:09 .mozilla
```

At this point, a locked account called **juan** exists on the system. To activate it, the administrator must next assign a password to the account using the **passwd** command and, optionally, set password aging guidelines.

3.3.2. Adding a New Group

To add a new group to the system, type the following at a shell prompt as **root**:

```
groupadd [options] group_name
```

...where *options* are command line options as described in [Table 3.3, “groupadd command line options”](#).

Table 3.3. groupadd command line options

Option	Description
-f, --force	When used with -g gid and <i>gid</i> already exists, groupadd will choose another unique <i>gid</i> for the group.
-g gid	Group ID for the group, which must be unique and greater than 499.
-K, --key key=value	Override /etc/login.defs defaults.
-o, --non-unique	Allow to create groups with duplicate.
-p, --password password	Use this encrypted password for the new group.
-r	Create a system group with a GID less than 500.

3.3.3. Creating Group Directories

System administrators usually like to create a group for each major project and assign people to the group when they need to access that project's files. With this traditional scheme, file managing is difficult; when someone creates a file, it is associated with the primary group to which they belong. When a single person works on multiple projects, it becomes difficult to associate the right files with the right group. However, with the UPG scheme, groups are automatically assigned to files created within a directory with the *setgid* bit set. The *setgid* bit makes managing group projects that share a common directory very simple because any files a user creates within the directory are owned by the group which owns the directory.

For example, a group of people need to work on files in the `/opt/myproject/` directory. Some people are trusted to modify the contents of this directory, but not everyone.

1. As **root**, create the `/opt/myproject/` directory by typing the following at a shell prompt:

```
mkdir /opt/myproject
```

2. Add the **myproject** group to the system:

```
groupadd myproject
```

3. Associate the contents of the `/opt/myproject/` directory with the **myproject** group:

```
chown root:myproject /opt/myproject
```

4. Allow users to create files within the directory, and set the setgid bit:

```
chmod 2775 /opt/myproject
```

At this point, all members of the **myproject** group can create and edit files in the `/opt/myproject/` directory without the administrator having to change file permissions every time users write new files. To verify that the permissions have been set correctly, run the following command:

```
~]# ls -l /opt
total 4
drwxrwsr-x. 3 root myproject 4096 Mar 3 18:31 myproject
```

3.4. Additional Resources

For more information on how to manage users and groups on Red Hat Enterprise Linux, see the resources listed below.

Installed Documentation

For information about various utilities for managing users and groups, see the following manual pages:

- ▶ **useradd(8)** — The manual page for the **useradd** command documents how to use it to create new users.
- ▶ **userdel(8)** — The manual page for the **userdel** command documents how to use it to delete users.
- ▶ **usermod(8)** — The manual page for the **usermod** command documents how to use it to modify users.
- ▶ **groupadd(8)** — The manual page for the **groupadd** command documents how to use it to create new groups.
- ▶ **groupdel(8)** — The manual page for the **groupdel** command documents how to use it to delete groups.
- ▶ **groupmod(8)** — The manual page for the **groupmod** command documents how to use it to modify group membership.
- ▶ **gpasswd(1)** — The manual page for the **gpasswd** command documents how to manage the `/etc/group` file.

- ▶ **grpck(8)** — The manual page for the **grpck** command documents how to use it to verify the integrity of the **/etc/group** file.
- ▶ **pwck(8)** — The manual page for the **pwck** command documents how to use it to verify the integrity of the **/etc/passwd** and **/etc/shadow** files.
- ▶ **pwconv(8)** — The manual page for the **pwconv** command documents how to use it to convert standard passwords to shadow passwords.
- ▶ **pwunconv(8)** — The manual page for the **pwunconv** command documents how to use it to convert shadow passwords to standard passwords.

For information about related configuration files, see:

- ▶ **group(5)** — The manual page for the **/etc/group** file documents how to use this file to define system groups.
- ▶ **passwd(5)** — The manual page for the **/etc/passwd** file documents how to use this file to define user information.
- ▶ **shadow(5)** — The manual page for the **/etc/shadow** file documents how to use this file to set passwords and account expiration information for the system.

Online Documentation

- ▶ [Red Hat Enterprise Linux 7 Security Guide](#) — The *Security Guide* for Red Hat Enterprise Linux 7 provides additional information how to ensure password security and secure the workstation by enabling password aging and user account locking.

See Also

- ▶ [Chapter 4, Gaining Privileges](#) documents how to gain administrative privileges by using the **su** and **sudo** commands.

Chapter 4. Gaining Privileges

System administrators (and in some cases users) will need to perform certain tasks with administrative access. Accessing the system as **root** is potentially dangerous and can lead to widespread damage to the system and data. This chapter covers ways to gain administrative privileges using setuid programs such as **su** and **sudo**. These programs allow specific users to perform tasks which would normally be available only to the root user while maintaining a higher level of control and system security.

See the *Red Hat Enterprise Linux 7 Security Guide* for more information on administrative controls, potential dangers and ways to prevent data loss resulting from improper use of privileged access.

4.1. The su Command

When a user executes the **su** command, they are prompted for the root password and, after authentication, are given a root shell prompt.

Once logged in via the **su** command, the user *is* the root user and has absolute administrative access to the system [1]. In addition, once a user has become root, it is possible for them to use the **su** command to change to any other user on the system without being prompted for a password.

Because this program is so powerful, administrators within an organization may wish to limit who has access to the command.

One of the simplest ways to do this is to add users to the special administrative group called *wheel*. To do this, type the following command as root:

```
usermod -G wheel <username>
```

In the previous command, replace **<username>** with the username you want to add to the **wheel** group.

You can also use the **User Manager** to modify group memberships, as follows. Note: you need Administrator privileges to perform this procedure.

1. Click the **System** menu on the Panel, point to **Administration** and then click **Users and Groups** to display the User Manager. Alternatively, type the command **system-config-users** at a shell prompt.
2. Click the **Users** tab, and select the required user in the list of users.
3. Click **Properties** on the toolbar to display the User Properties dialog box (or choose **Properties** on the **File** menu).
4. Click the **Groups** tab, select the check box for the **wheel** group, and then click **OK**.

See [Section 3.2, “Using the User Manager Tool”](#) for more information about the **User Manager**.

After you add the desired users to the **wheel** group, it is advisable to only allow these specific users to use the **su** command. To do this, you will need to edit the PAM configuration file for **su**: **/etc/pam.d/su**. Open this file in a text editor and remove the comment (#) from the following line:

```
#auth      required      pam_wheel.so  use_uid
```

This change means that only members of the administrative group **wheel** can switch to another user using the **su** command.

**Note**

The **root** user is part of the **wheel** group by default.

4.2. The sudo Command

The **sudo** command offers another approach to giving users administrative access. When trusted users precede an administrative command with **sudo**, they are prompted for *their own* password. Then, when they have been authenticated and assuming that the command is permitted, the administrative command is executed as if they were the root user.

The basic format of the **sudo** command is as follows:

```
sudo <command>
```

In the above example, **<command>** would be replaced by a command normally reserved for the root user, such as **mount**.

The **sudo** command allows for a high degree of flexibility. For instance, only users listed in the **/etc/sudoers** configuration file are allowed to use the **sudo** command and the command is executed in *the user's shell*, not a root shell. This means the root shell can be completely disabled as shown in the *Red Hat Enterprise Linux 7 Security Guide*.

Each successful authentication using the **sudo** is logged to the file **/var/log/messages** and the command issued along with the issuer's username is logged to the file **/var/log/secure**. Should you require additional logging, use the **pam_tty_audit** module to enable TTY auditing for specified users by adding the following line to your **/etc/pam.d/system-auth** file:

```
session required pam_tty_audit.so disable=<pattern> enable=<pattern>
```

where *pattern* represents a comma-separated listing of users with an optional use of globs. For example, the following configuration will enable TTY auditing for the root user and disable it for all other users:

```
session required pam_tty_audit.so disable=* enable=root
```

Another advantage of the **sudo** command is that an administrator can allow different users access to specific commands based on their needs.

Administrators wanting to edit the **sudo** configuration file, **/etc/sudoers**, should use the **visudo** command.

To give someone full administrative privileges, type **visudo** and add a line similar to the following in the user privilege specification section:

```
juan ALL=(ALL) ALL
```

This example states that the user, **juan**, can use **sudo** from any host and execute any command.

The example below illustrates the granularity possible when configuring **sudo**:

```
%users localhost=/sbin/shutdown -h now
```

This example states that any user can issue the command `/sbin/shutdown -h now` as long as it is issued from the console.

The man page for **sudoers** has a detailed listing of options for this file.



Important

There are several potential risks to keep in mind when using the **sudo** command. You can avoid them by editing the **/etc/sudoers** configuration file using **visudo** as described above. Leaving the **/etc/sudoers** file in its default state gives every user in the **wheel** group unlimited root access.

- ▶ By default, **sudo** stores the sudoer's password for a five minute timeout period. Any subsequent uses of the command during this period will not prompt the user for a password. This could be exploited by an attacker if the user leaves his workstation unattended and unlocked while still being logged in. This behavior can be changed by adding the following line to the **/etc/sudoers** file:

```
Defaults    timestamp_timeout=<value>
```

where `<value>` is the desired timeout length in minutes. Setting the `<value>` to 0 causes **sudo** to require a password every time.

- ▶ If a sudoer's account is compromised, an attacker can use **sudo** to open a new shell with administrative privileges:

```
sudo /bin/bash
```

Opening a new shell as root in this or similar fashion gives the attacker administrative access for a theoretically unlimited amount of time, bypassing the timeout period specified in the **/etc/sudoers** file and never requiring the attacker to input a password for **sudo** again until the newly opened session is closed.

4.3. Additional Resources

While programs allowing users to gain administrative privileges are a potential security risk, security itself is beyond the scope of this particular book. You should therefore refer to the resources listed below for more information regarding security and privileged access.

Installed Documentation

- ▶ **su(1)** — The manual page for **su** provides information regarding the options available with this command.
- ▶ **sudo(8)** — The manual page for **sudo** includes a detailed description of this command lists options available for customizing its behavior.
- ▶ **pam(8)** — The manual page describing the use of Pluggable Authentication Modules (PAM) for Linux.

Online Documentation

- ▶ [Red Hat Enterprise Linux 7 Security Guide](#) — The *Security Guide* for Red Hat Enterprise Linux 7 provides a more in-depth look at potential security issues pertaining to setuid programs as well as techniques used to alleviate these risks.

See Also

- ▶ [Chapter 3, Managing Users and Groups](#) documents how to manage system users and groups in the graphical user interface and on the command line.

[1] This access is still subject to the restrictions imposed by SELinux, if it is enabled.

Part II. Package Management

All software on a Red Hat Enterprise Linux system is divided into RPM packages, which can be installed, upgraded, or removed. This part describes how to manage packages on Red Hat Enterprise Linux using **Yum**.

Chapter 5. Yum

Yum is the Red Hat package manager that is able to query for information about available packages, fetch packages from repositories, install and uninstall them, and update an entire system to the latest available version. Yum performs automatic dependency resolution on packages you are updating, installing, or removing, and thus is able to automatically determine, fetch, and install all available dependent packages.

Yum can be configured with new, additional repositories, or *package sources*, and also provides many plug-ins which enhance and extend its capabilities. Yum is able to perform many of the same tasks that **RPM** can; additionally, many of the command line options are similar. Yum enables easy and simple package management on a single machine or on groups of them.

The following sections assume your system was registered with Red Hat Subscription Management during installation as described in the [Red Hat Enterprise Linux 7 Installation Guide](#). For additional information, see [Using and Configuring Red Hat Subscription Manager](#).



Important

Yum provides secure package management by enabling GPG (Gnu Privacy Guard; also known as GnuPG) signature verification on GPG-signed packages to be turned on for all package repositories (i.e. package sources), or for individual repositories. When signature verification is enabled, Yum will refuse to install any packages not GPG-signed with the correct key for that repository. This means that you can trust that the **RPM** packages you download and install on your system are from a trusted source, such as Red Hat, and were not modified during transfer. See [Section 5.5, “Configuring Yum and Yum Repositories”](#) for details on enabling signature-checking with Yum, or [Section A.3.2, “Checking Package Signatures”](#) for information on working with and verifying GPG-signed **RPM** packages in general.

Yum also enables you to easily set up your own repositories of **RPM** packages for download and installation on other machines. When possible, Yum uses *parallel download* of multiple packages and metadata to speed up downloading.

Learning Yum is a worthwhile investment because it is often the fastest way to perform system administration tasks, and it provides capabilities beyond those provided by the **PackageKit** graphical package management tools.



Note

You must have superuser privileges in order to use **yum** to install, update or remove packages on your system. All examples in this chapter assume that you have already obtained superuser privileges by using either the **su** or **sudo** command.

5.1. Checking For and Updating Packages

Yum allows you to check if your system has any updates that wait to be applied. You can list packages that need to be updated and update them as a whole, or you can update a selected individual package.

5.1.1. Checking For Updates

To see which installed packages on your system have updates available, use the following command:

```
yum check-update
```

Example 5.1. Example output of the yum check-update command

The output of **yum check-update** can look as follows:

```
~]# yum check-update
Loaded plugins: langpacks, product-id, subscription-manager
Updating Red Hat repositories.
INFO:rhsm-app.repolib:repos updated: 0
PackageKit.x86_64          0.5.8-2.el6      rhel
PackageKit-glib.x86_64       0.5.8-2.el6      rhel
PackageKit-yum.x86_64        0.5.8-2.el6      rhel
PackageKit-yum-plugin.x86_64 0.5.8-2.el6      rhel
glibc.x86_64                2.11.90-20.el6   rhel
glibc-common.x86_64         2.10.90-22     rhel
kernel.x86_64               2.6.31-14.el6   rhel
rpm.x86_64                 4.7.1-5.el6     rhel
rpm-libs.x86_64             4.7.1-5.el6     rhel
rpm-python.x86_64           4.7.1-5.el6     rhel
yum.noarch                  3.2.24-4.el6    rhel
```

The packages in the above output are listed as having updates available. The first package in the list is **PackageKit**, the graphical package manager. The line in the example output tells us:

- » **PackageKit** — the name of the package
- » **x86_64** — the CPU architecture the package was built for
- » **0.5.8** — the version of the updated package to be installed
- » **rhel** — the repository in which the updated package is located

The output also shows us that we can update the kernel (the *kernel* package), Yum and RPM themselves (the *yum* and *rpm* packages), as well as their dependencies (such as the *rpm-libs*, and *rpm-python* packages), all using **yum**.

5.1.2. Updating Packages

You can choose to update a single package, multiple packages, or all packages at once. If any dependencies of the package (or packages) you update have updates available themselves, then they are updated too.

Updating a Single Package

To update a single package, run the following command as **root**:

```
yum update package_name
```

Example 5.2. Updating the rpm package

To update the *rpm* package, type:

```
~]# yum update rpm
Loaded plugins: langpacks, product-id, subscription-manager
Updating Red Hat repositories.
INFO:rhsm-app.repolib:repos updated: 0
```

```

Setting up Update Process
Resolving Dependencies
--> Running transaction check
----> Package rpm.x86_64 0:4.11.1-3.el7 will be updated
--> Processing Dependency: rpm = 4.11.1-3.el7 for package: rpm-libs-4.11.1-
3.el7.x86_64
--> Processing Dependency: rpm = 4.11.1-3.el7 for package: rpm-python-4.11.1-
3.el7.x86_64
--> Processing Dependency: rpm = 4.11.1-3.el7 for package: rpm-build-4.11.1-
3.el7.x86_64
----> Package rpm.x86_64 0:4.11.2-2.el7 will be an update
--> Running transaction check
...
--> Finished Dependency Resolution

Dependencies Resolved
=====


| Package                           | Arch   | Version      | Repository | Size  |
|-----------------------------------|--------|--------------|------------|-------|
| <b>Updating:</b>                  |        |              |            |       |
| rpm                               | x86_64 | 4.11.2-2.el7 | rhel       | 1.1 M |
| <b>Updating for dependencies:</b> |        |              |            |       |
| rpm-build                         | x86_64 | 4.11.2-2.el7 | rhel       | 139 k |
| rpm-build-libs                    | x86_64 | 4.11.2-2.el7 | rhel       | 98 k  |
| rpm-libs                          | x86_64 | 4.11.2-2.el7 | rhel       | 261 k |
| rpm-python                        | x86_64 | 4.11.2-2.el7 | rhel       | 74 k  |


Transaction Summary
=====
Upgrade 1 Package (+4 Dependent packages)

Total size: 1.7 M
Is this ok [y/d/N]:

```

This output contains several items of interest:

1. **Loaded plugins:** `langpacks`, `product-id`, `subscription-manager` — `yum` always informs you which Yum plug-ins are installed and enabled. See [Section 5.6, “Yum Plug-ins”](#) for general information on Yum plug-ins, or to [Section 5.6.3, “Working with Yum Plug-ins”](#) for descriptions of specific plug-ins.
2. **`rpm.x86_64`** — you can download and install new `rpm` package as well as its dependencies. Transaction check is performed for each of these packages.
3. **`yum`** presents the update information and then prompts you as to whether you want it to perform the update; `yum` runs interactively by default. If you already know which transactions the `yum` command plans to perform, you can use the `-y` option to automatically answer `yes` to any questions that `yum` asks (in which case it runs non-interactively). However, you should always examine which changes `yum` plans to make to the system so that you can easily troubleshoot any problems that might arise. You can also choose to download the package without installing it. To do so, select the `d` option at the download prompt. This launches a background download of the selected package.

If a transaction does go awry, you can view Yum transaction history by using the `yum history` command as described in [Section 5.4, “Working with Transaction History”](#).



Important

yum always *installs* a new kernel in the same sense that **RPM** installs a new kernel when you use the command **rpm -i kernel**. Therefore, you do not need to worry about the distinction between *installing* and *upgrading* a kernel package when you use **yum**: it will do the right thing, regardless of whether you are using the **yum update** or **yum install** command.

When using **RPM**, on the other hand, it is important to use the **rpm -i kernel** command (which installs a new kernel) instead of **rpm -u kernel** (which *replaces* the current kernel). See [Section A.2.1, “Installing and Upgrading Packages”](#) for more information on installing/upgrading kernels with **RPM**.

Similarly, it is possible to update a package group. Type as **root**:

```
yum group update group_name
```

Here, replace *group_name* with a name of the package group you wish to update. For more information on package groups, see [Section 5.3, “Working with Package Groups”](#).

Yum also offers the **upgrade** command that is equal to **update** with enabled **obsoletes** configuration option (see [Section 5.5.1, “Setting \[main\] Options”](#)). By default, **obsoletes** is turned on in **/etc/yum.conf**, which makes these two commands equivalent.

Updating All Packages and Their Dependencies

To update all packages and their dependencies, simply enter **yum update** (without any arguments):

```
yum update
```

Updating Security-Related Packages

Discovering which packages have security updates available and then updating those packages quickly and easily is important. Yum provides the plug-in for this purpose. The **security** plug-in extends the **yum** command with a set of highly-useful security-centric commands, subcommands and options. See [Section 5.6.3, “Working with Yum Plug-ins”](#) for specific information.

5.1.3. Preserving Configuration File Changes

You will inevitably make changes to the configuration files installed by packages as you use your Red Hat Enterprise Linux system. **RPM**, which Yum uses to perform changes to the system, provides a mechanism for ensuring their integrity. See [Section A.2.1, “Installing and Upgrading Packages”](#) for details on how to manage changes to configuration files across package upgrades.

5.2. Working with Packages

Yum allows you to perform a complete set of operations with software packages, including searching for packages, viewing information about them, installing and removing.

5.2.1. Searching Packages

You can search all RPM package names, descriptions and summaries by using the following command:

```
yum search term...
```

This command displays the list of matches for each term.

Example 5.3. Searching for packages matching a specific string

To list all packages that match “meld” or “kompare”, type:

```
~]$ yum search meld kompare
Loaded plugins: langpacks, langpacks, product-id, subscription-manager
Updating Red Hat repositories.
INFO:rhsm-app.repolib:repos updated: 0
=====
N/S matched: kompare =====
kompare.x86_64 : Diff tool
...
Name and summary matches mostly, use "search all" for everything.
Warning: No matches found for: meld
```

The **yum search** command is useful for searching for packages you do not know the name of, but for which you know a related term. Note that by default, **yum search** returns matches in package name and summary, which makes the search faster. Use the **yum search all** command for more exhaustive but slower search.

Filtering the Results

All of Yum’s list commands allow you to filter the results by appending one or more *glob expressions* as arguments. Glob expressions are normal strings of characters which contain one or more of the wildcard characters * (which expands to match any character subset) and ? (which expands to match any single character).

Be careful to escape the glob expressions when passing them as arguments to a **yum** command, otherwise the Bash shell will interpret these expressions as *pathname expansions*, and potentially pass all files in the current directory that match the global expressions to **yum**. To make sure the glob expressions are passed to **yum** as intended, either:

- ▶ escape the wildcard characters by preceding them with a backslash character
- ▶ double-quote or single-quote the entire glob expression.

Examples in the following section demonstrate usage of both these methods.

5.2.2. Listing Packages

To list information on all installed and available packages type the following at a shell prompt:

```
yum list all
```

To list installed and available packages that match inserted glob expressions use the following command:

```
yum list glob_expression...
```

Example 5.4. Listing ABRT-related packages

Packages with various ABRT add-ons and plug-ins either begin with “abrt-addon-”, or “abrt-plugin-”. To list these packages, type the following command at a shell prompt. Note how the wildcard characters are escaped with a backslash character:

```
~]$ yum list abrt-addon\* abrt-plugin\*
Loaded plugins: langpacks, product-id, subscription-manager
Updating Red Hat repositories.
INFO:rhsm-app.repolib:repos updated: 0
Installed Packages
abrt-addon-ccpp.x86_64          1.0.7-5.el6      @rhel
abrt-addon-kerneloops.x86_64     1.0.7-5.el6      @rhel
abrt-addon-python.x86_64         1.0.7-5.el6      @rhel
abrt-plugin-bugzilla.x86_64      1.0.7-5.el6      @rhel
abrt-plugin-logger.x86_64        1.0.7-5.el6      @rhel
abrt-plugin-sosreport.x86_64     1.0.7-5.el6      @rhel
abrt-plugin-ticketuploader.x86_64 1.0.7-5.el6      @rhel
```

To list all packages installed on your system use the **installed** keyword. The rightmost column in the output lists the repository from which the package was retrieved.

```
yum list installed glob_expression...
```

Example 5.5. Listing all installed versions of the krb package

The following example shows how to list all installed packages that begin with “krb” followed by exactly one character and a hyphen. This is useful when you want to list all versions of certain component as these are distinguished by numbers. The entire glob expression is quoted to ensure proper processing.

```
~]$ yum list installed "krb?-*"
Loaded plugins: langpacks, product-id, subscription-manager
Updating Red Hat repositories.
INFO:rhsm-app.repolib:repos updated: 0
Installed Packages
krb5-libs.x86_64                1.8.1-3.el6      @rhel
krb5-workstation.x86_64           1.8.1-3.el6      @rhel
```

To list all packages in all enabled repositories that are available to install, use the command in the following form:

```
yum list available glob_expression...
```

Example 5.6. Listing available gstreamer plug-ins

For instance, to list all available packages with names that contain “gstreamer” and then “plugin”, run the following command:

```
~]$ yum list available gstreamer\*plugin\*
Loaded plugins: langpacks, product-id, subscription-manager
Updating Red Hat repositories.
INFO:rhsm-app.repolib:repos updated: 0
Available Packages
gstreamer-plugins-bad-free.i686           0.10.17-4.el6      rhel
gstreamer-plugins-base.i686                0.10.26-1.el6      rhel
gstreamer-plugins-base-devel.i686          0.10.26-1.el6      rhel
gstreamer-plugins-base-devel.x86_64        0.10.26-1.el6      rhel
gstreamer-plugins-good.i686               0.10.18-1.el6      rhel
```

Listing Repositories

To list the repository ID, name, and number of packages for each *enabled* repository on your system, use the following command:

```
yum repolist
```

To list more information about these repositories, add the **-v** option. With this option enabled, information including the file name, overall size, date of the last update, and base URL are displayed for each listed repository. As an alternative, you can use the **repoinfo** command that produces the same output.

```
yum repolist -v
```

```
yum repoinfo
```

To list both enabled and disabled repositories use the following command. A status column is added to the output list to show which of the repositories are enabled.

```
yum repolist all
```

By passing **disabled** as a first argument, you can reduce the command output to disabled repositories. For further specification you can pass the ID or name of repositories or related glob_expressions as arguments. Note that if there is an exact match between the repository ID or name and the inserted argument, this repository is listed even if it does not pass the *enabled* or *disabled* filter.

5.2.3. Displaying Package Information

To display information about one or more packages, use the following command (glob expressions are valid here as well):

```
yum info package_name...
```

Replace *package_name* with the name of the package.

Example 5.7. Displaying information on the abrt package

To display information about the *abrt* package, type:

```
~]$ yum info abrt
Loaded plugins: langpacks, product-id, subscription-manager
Updating Red Hat repositories.
INFO:rhsm-app.repolib:repos updated: 0
Installed Packages
Name        : abrt
Arch       : x86_64
Version    : 1.0.7
Release    : 5.el6
Size       : 578 k
Repo       : installed
From repo : rhel
Summary    : Automatic bug detection and reporting tool
URL        : https://fedorahosted.org/abrt/
License    : GPLv2+
Description: abrt is a tool to help users to detect defects in applications
             : and to create a bug report with all informations needed by
             : maintainer to fix it. It uses plugin system to extend its
             : functionality.
```

The **yum info package_name** command is similar to the **rpm -q --info package_name** command, but provides as additional information the ID of the Yum repository the RPM package is found in (look for the **From repo:** line in the output).

Using yumdb

You can also query the Yum database for alternative and useful information about a package by using the following command:

```
yumdb info package_name
```

This command provides additional information about a package, including the checksum of the package (and algorithm used to produce it, such as SHA-256), the command given on the command line that was invoked to install the package (if any), and the reason that the package is installed on the system (where **user** indicates it was installed by the user, and **dep** means it was brought in as a dependency).

Example 5.8. Querying yumdb for information on the yum package

To display additional information about the `yum` package, type:

```
~]$ yumdb info yum
Loaded plugins: langpacks, product-id, subscription-manager
yum -3.2.27-4.el6.noarch
checksum_data =
23d337ed51a9757bbfbcdceb82c4eaca9808ff1009b51e9626d540f44fe95f771
checksum_type = sha256
from_repo = rhel
from_repo_revision = 1298613159
from_repo_timestamp = 1298614288
installed_by = 4294967295
reason = user
releasever = 6.1
```

For more information on the `yumdb` command, see the `yumdb(8)` manual page.

5.2.4. Installing Packages

To install a single package and all of its non-installed dependencies, enter a command in the following form (as `root`):

```
yum install package_name
```

You can also install multiple packages simultaneously by appending their names as arguments. To do so, type as `root`:

```
yum install package_name package_name...
```

If you are installing packages on a *multilib* system, such as an AMD64 or Intel64 machine, you can specify the architecture of the package (as long as it is available in an enabled repository) by appending `.arch` to the package name:

```
yum install package_name.arch
```

Example 5.9. Installing packages on multilib system

To install the `sqlite` package for the `i686` architecture, type:

```
~]# yum install sqlite.i686
```

You can use glob expressions to quickly install multiple similarly-named packages. As `root`:

```
yum install glob_expression...
```

Example 5.10. Installing all audacious plugins

Global expressions are useful when you want to install several packages with similar names. To install all audacious plug-ins, use the command in the following form:

```
~]# yum install audacious-plugins-\*
```

In addition to package names and glob expressions, you can also provide file names to **yum install**. If you know the name of the binary you want to install, but not its package name, you can give **yum install** the path name. As **root**, type:

```
yum install /usr/sbin/named
```

yum then searches through its package lists, finds the package which provides **/usr/sbin/named**, if any, and prompts you as to whether you want to install it.

As you can see in the above examples, the **yum install** command does not require strictly defined arguments. It can process various formats of package names and glob expressions, which makes installation easier for users. On the other hand, it takes some time till **yum** parses the input correctly, especially if you specify a large number of packages. To optimize the package search, you can use the following commands to explicitly define how to parse the arguments:

```
yum install-n name
```

```
yum install-na name.architecture
```

```
yum install-nevra name-epoch:version-release.architecture
```

With **install-n**, **yum** interprets *name* as exact name of the package. The **install-na** command tells **yum** that the subsequent argument contains the package name and architecture divided by the dot character. With **install-nevra**, **yum** will expect argument in the form *name-epoch:version-release.architecture*. Similarly, you can use **yum remove-n**, **yum remove-na**, and **yum remove-nevra** when searching for packages to be removed.

**Note**

If you know you want to install the package that contains the **named** binary, but you do not know in which **bin** or **sbin** directory is the file installed, use the **yum provides** command with a glob expression:

```
~]# yum provides "*bin/named"
Loaded plugins: langpacks, product-id, subscription-manager
Updating Red Hat repositories.
INFO:rhsms-app.repolib/repos updated: 0
32:bind-9.7.0-4.P1.el6.x86_64 : The Berkeley Internet Name Domain (BIND)
                                     : DNS (Domain Name System) server
Repo        : rhel
Matched from:
Filename    : /usr/sbin/named
```

yum provides "*/file_name" is a common and useful trick to find the package(s) that contain *file_name*.

Example 5.11. Installation Process

The following example provides an overview of installation with use of **yum**. Imagine you want to download and install the latest version of the *httpd* package. To do so, execute as **root**:

```
~]# yum install httpd
Loaded plugins: langpacks, product-id, subscription-manager
Resolving Dependencies
--> Running transaction check
----> Package httpd.x86_64 0:2.4.6-12.el7 will be updated
----> Package httpd.x86_64 0:2.4.6-13.el7 will be an update
--> Processing Dependency: 2.4.6-13.el7 for package: httpd-2.4.6-13.el7.x86_64
--> Running transaction check
----> Package httpd-tools.x86_64 0:2.4.6-12.el7 will be updated
----> Package httpd-tools.x86_64 0:2.4.6-13.el7 will be an update
--> Finished Dependency Resolution

Dependencies Resolved
```

After executing the above command, **yum** loads the necessary plug-ins and runs the transaction check. In this case, *httpd* is already installed. Since the installed package is older than the latest currently available version, it will be updated. The same applies to the *httpd-tools* package that *httpd* depends on. Then, a transaction summary is displayed:

```
=====
=
Package      Arch      Version       Repository      Size
=====
=
Updating:
 httpd        x86_64    2.4.6-13.el7   rhel-x86_64-server-7  1.2 M
Updating for dependencies:
 httpd-tools  x86_64    2.4.6-13.el7   rhel-x86_64-server-7  77 k

Transaction Summary
```

```
=====
=
Upgrade 1 Package (+1 Dependent package)

Total size: 1.2 M
Is this ok [y/d/N]:
```

In this step **yum** prompts you to confirm the installation. Apart from **y** (yes) and **N** (no) options, you can choose **d** (download only) to download the packages but not to install them directly. If you choose **y**, the installation proceeds with the following messages until it is finished successfully.

```
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Updating : httpd-tools-2.4.6-13.el7.x86_64          1/4
  Updating : httpd-2.4.6-13.el7.x86_64                2/4
  Cleanup   : httpd-2.4.6-12.el7.x86_64              3/4
  Cleanup   : httpd-tools-2.4.6-12.el7.x86_64         4/4
  Verifying : httpd-2.4.6-13.el7.x86_64              1/4
  Verifying : httpd-tools-2.4.6-13.el7.x86_64         2/4
  Verifying : httpd-tools-2.4.6-12.el7.x86_64         3/4
  Verifying : httpd-2.4.6-12.el7.x86_64               4/4

Updated:
httpd.x86_64 0:2.4.6-13.el7

Dependency Updated:
httpd-tools.x86_64 0:2.4.6-13.el7

Complete!
```

To install a previously-downloaded package from the local directory on your system, use the following command:

```
yum localinstall path
```

Replace *path* with a path to the package you wish to install.

5.2.5. Downloading Packages

As shown in [Example 5.11, “Installation Process”](#), at certain point of installation process you are prompted to confirm the installation with the following message:

```
...
Total size: 1.2 M
Is this ok [y/d/N]:
...
```

By choosing the **d** option, you tell **yum** to download the packages without installing them immediately. You can install these packages later in off-line mode with the **yum localinstall** command or you can share them with a different device. Downloaded packages are saved in one of the subdirectories of the cache directory, by default

/var/cache/yum/\$basearch/\$releasever/packages/ directory. The downloading proceeds in background mode so that you can use **yum** for other operations in parallel.

5.2.6. Removing Packages

Similarly to package installation, Yum allows you to uninstall (remove in **RPM** and Yum terminology) them. To uninstall a particular package, as well as any packages that depend on it, run the following command as **root**:

```
yum remove package_name...
```

As when you install multiple packages, you can remove several at once by adding more package names to the command.

Example 5.12. Removing several packages

To remove *totem*, type the following at a shell prompt:

```
~]# yum remove totem
```

Similar to **install**, **remove** can take these arguments:

- ▶ package names
- ▶ glob expressions
- ▶ file lists
- ▶ package provides



Warning

Yum is not able to remove a package without also removing packages which depend on it. This type of operation can only be performed by **RPM**, is not advised, and can potentially leave your system in a non-functioning state or cause applications to misbehave and/or crash. For further information, see [Section A.2.2, “Uninstalling Packages”](#) in the **RPM** chapter.

5.3. Working with Package Groups

A package group is a collection of packages that serve a common purpose, for instance *System Tools* or *Sound and Video*. Installing a package group pulls a set of dependent packages, saving time considerably. The **yum groups** command is a top-level command that covers all the operations that act on package groups in Yum.

5.3.1. Listing Package Groups

The **summary** option is used to view the number of installed groups, available groups, available environment groups, and both installed and available language groups:

```
yum groups summary
```

Example 5.13. Example output of yum groups summary

```
~]$ yum groups summary
Loaded plugins: langpacks, product-id, subscription-manager
Available Environment Groups: 12
Installed Groups: 10
Available Groups: 12
```

To list all package groups from yum repositories add the **list** option. You can filter the command output by group names.

```
yum group list glob_expression...
```

Several optional arguments can be passed to this command, including **hidden** to list also groups not marked as user visible, and **ids** to list group IDs. You can add **language**, **environment**, **installed**, or **available** options to reduce the command output to specific group type.

To list mandatory and optional packages contained in a particular group, use the following command:

```
yum group info glob_expression...
```

Example 5.14. Viewing information on the LibreOffice package group

```
~]$ yum group info LibreOffice
Loaded plugins: langpacks, product-id, subscription-manager

Group: LibreOffice
Group-Id: libreoffice
Description: LibreOffice Productivity Suite
Mandatory Packages:
=libreoffice-calc
libreoffice-draw
-libreoffice-emailmerge
libreoffice-graphicfilter
=libreoffice-impress
=libreoffice-math
=libreoffice-writer
+libreoffice-xsltfilter
Optional Packages:
libreoffice-base
libreoffice-pyuno
```

As you can see in the above example, the packages included in the package group can have different states that are marked with the following symbols:

- ▶ " - " — Package is not installed and it will not be installed as a part of the package group.
- ▶ " + " — Package is not installed but it will be installed on next **yum upgrade** or **yum group upgrade**.
- ▶ " = " — Package is installed and it was installed as a part of the package group.
- ▶ *no symbol* — Package is installed but it was installed outside of the package group. This means that the **yum group remove** will not remove these packages.

These distinctions take place only when the **group_command** configuration parameter is set to **objects**, which is the default setting. Set this parameter to a different value if you do not want yum to track if a package was installed as a part of the group or separately, which will make "no symbol" packages equivalent to "=" packages.

You can alter the above package states with use of the **yum group mark** command. For example, **yum group mark packages** marks any given installed packages as members of a specified group. To avoid installation of new packages on group update, use **yum group mark blacklist**. See **yum** man page for more information on capabilities of **yum group mark**.

Note

You can identify an environmental group with use of the @^ prefix and a package group can be marked with @. When using **yum group list**, **info**, **install**, or **remove**, pass **@group_name** to specify a package group, **@^group_name** to specify an environmental group, or **group_name** to include both.

5.3.2. Installing a Package Group

Each package group has a name and a *groupid*. To list the names of all package groups, and their groupids, which are displayed in parentheses, type:

```
yum group list ids
```

Example 5.15. Finding name and groupid of a package group

Imagine you want to install a package group related to the KDE desktop environment, but you cannot remember the exact name or id of the package group. To find the information, type:

```
~]$ yum group list ids kde\*
Loaded plugins: langpacks, product-id, subscription-manager
Available Groups:
  KDE Desktop (kde-desktop)
Done
```

You can install a package group by passing its full group name (without the groupid part) to the **group install** command. As **root**, type:

```
yum group install group_name
```

You can also install by groupid. As **root**, execute the following command:

```
yum group install groupid
```

You can pass the groupid or quoted name to the **install** command if you prepend it with an @-symbol, which tells **yum** that you want to perform **group install**. As **root**, type:

```
yum install @group
```

Replace **group** with the groupid or quoted group name. Similar logic applies to environmental groups:

```
yum install @^group
```

Example 5.16. Four equivalent ways of installing the KDE Desktop group

As mentioned before, you can use four alternative, but equivalent ways to install a package group. For KDE Desktop, the commands look as follows:

```
~]# yum group install "KDE Desktop"
~]# yum group install kde-desktop
~]# yum install @"KDE Desktop"
~]# yum install @kde-desktop
```

5.3.3. Removing a Package Group

You can remove a package group using syntax congruent with the **install** syntax, with use of either name of the package group or its id. As **root**, type:

```
yum group remove group_name
```

```
yum group remove groupid
```

Also, you can pass the groupid or quoted name to the **remove** command if you prepend it with an @-symbol, which tells **yum** that you want to perform **group remove**. As **root**, type:

```
yum remove @group
```

Replace *group* with the groupid or quoted group name. Similarly, you can replace an environmental group:

```
yum remove @^group
```

Example 5.17. Four equivalent ways of removing the KDE Desktop group

Similarly to install, you can use four alternative, but equivalent ways to remove a package group. For KDE Desktop, the commands look as follows:

```
~]# yum group remove "KDE Desktop"
~]# yum group remove kde-desktop
~]# yum remove @"KDE Desktop"
~]# yum remove @kde-desktop
```

5.4. Working with Transaction History

The **yum history** command allows users to review information about a timeline of Yum transactions, the dates and times they occurred, the number of packages affected, whether transactions succeeded or were aborted, and if the RPM database was changed between transactions. Additionally, this command can be used to undo or redo certain transactions. All history data are stored in the *history DB in the /var/lib/yum/history/ directory*.

5.4.1. Listing Transactions

To display a list of twenty most recent transactions, as **root**, either run **yum history** with no additional arguments, or type the following at a shell prompt:

```
yum history list
```

To display all transactions, add the **all** keyword:

```
yum history list all
```

To display only transactions in a given range, use the command in the following form:

```
yum history list start_id..end_id
```

You can also list only transactions regarding a particular package or packages. To do so, use the command with a package name or a glob expression:

```
yum history list glob_expression...
```

Example 5.18. Listing the five oldest transactions

In the output of **yum history list**, the most recent transaction is displayed at the top of the list. To display information about the five oldest transactions stored in the history data base, type:

```
~]# yum history list 1..5
Loaded plugins: langpacks, product-id, subscription-manager
ID      | Login user           | Date and time   | Action(s)    | Altered
-----+
 5 | Jaromir ... <jhradilek> | 2013-07-29 15:33 | Install     | 1
 4 | Jaromir ... <jhradilek> | 2013-07-21 15:10 | Install     | 1
 3 | Jaromir ... <jhradilek> | 2013-07-16 15:27 | I, U        | 73
 2 | System <unset>         | 2013-07-16 15:19 | Update      | 1
 1 | System <unset>         | 2013-07-16 14:38 | Install     | 1106
history list
```

All forms of the **yum history list** command produce tabular output with each row consisting of the following columns:

- ▶ **ID** — an integer value that identifies a particular transaction.
- ▶ **Login user** — the name of the user whose login session was used to initiate a transaction. This information is typically presented in the **Full Name <username>** form. For transactions that were not issued by a user (such as an automatic system update), **System <unset>** is used instead.
- ▶ **Date and time** — the date and time when a transaction was issued.
- ▶ **Action(s)** — a list of actions that were performed during a transaction as described in [Table 5.1, “Possible values of the Action\(s\) field”](#).
- ▶ **Altered** — the number of packages that were affected by a transaction, possibly followed by additional information as described in [Table 5.2, “Possible values of the Altered field”](#).

Table 5.1. Possible values of the Action(s) field

Action	Abbreviation	Description
Downgrade	D	At least one package has been downgraded to an older version.
Erase	E	At least one package has been removed.
Install	I	At least one new package has been installed.
Obsoleting	O	At least one package has been marked as obsolete.
Reinstall	R	At least one package has been reinstalled.
Update	U	At least one package has been updated to a newer version.

Table 5.2. Possible values of the Altered field

Symbol	Description
<	Before the transaction finished, the rpmb database was changed outside Yum.
>	After the transaction finished, the rpmb database was changed outside Yum.
*	The transaction failed to finish.
#	The transaction finished successfully, but yum returned a non-zero exit code.
E	The transaction finished successfully, but an error or a warning was displayed.
P	The transaction finished successfully, but problems already existed in the rpmb database.
S	The transaction finished successfully, but the --skip-broken command line option was used and certain packages were skipped.

To synchronize the **rpmb** or **yumb** database contents for any installed package with the currently used **rpmb** or **yumb** database, type the following:

```
yum history sync
```

To display some overall statistics about the currently used history DB use the following command:

```
yum history stats
```

Example 5.19. Example output of **yum history stats**

```
~]# yum history stats
Loaded plugins: langpacks, product-id, subscription-manager
File        : //var/lib/yum/history/history-2012-08-15.sqlite
Size       : 2,766,848
Transactions: 41
Begin time  : Wed Aug 15 16:18:25 2012
End time    : Wed Feb 27 14:52:30 2013
Counts      :
  NEVRAC   : 2,204
  NEVRA    : 2,204
  NA       : 1,759
  NEVR     : 2,204
  rpm DB   : 2,204
  yum DB   : 2,204
history stats
```

Yum also allows you to display a summary of all past transactions. To do so, run the command in the following form as **root**:

```
yum history summary
```

To display only transactions in a given range, type:

```
yum history summary start_id..end_id
```

Similarly to the **yum history list** command, you can also display a summary of transactions regarding a certain package or packages by supplying a package name or a glob expression:

```
yum history summary glob_expression...
```

Example 5.20. Summary of the five latest transactions

```
~]# yum history summary 1..5
Loaded plugins: langpacks, product-id, subscription-manager
Login user           | Time                | Action(s)      | Altered
-----
Jaromir ... <jhradilek> | Last day          | Install        | 1
Jaromir ... <jhradilek> | Last week         | Install        | 1
Jaromir ... <jhradilek> | Last 2 weeks     | I, U          | 73
System <unset>          | Last 2 weeks     | I, U          | 1107
history summary
```

All forms of the **yum history summary** command produce simplified tabular output similar to the output of **yum history list**.

As shown above, both **yum history list** and **yum history summary** are oriented towards transactions, and although they allow you to display only transactions related to a given package or packages, they lack important details, such as package versions. To list transactions from the perspective of a package, run the following command as **root**:

```
yum history package-list glob_expression...
```

Example 5.21. Tracing the history of a package

For example, to trace the history of *subscription-manager* and related packages, type the following at a shell prompt:

```
~]# yum history package-list subscription-manager\*
Loaded plugins: langpacks, product-id, subscription-manager
ID      | Action(s)      | Package
-----
3 | Updated       | subscription-manager-1.10.11-1.el6.x86_64
3 | Update        | 1.10.17-1.el6_1.x86_64
3 | Updated       | subscription-manager-firstboot-1.10.11-1.el6.x86_64
3 | Update        | 1.10.17-1.el6_1.x86_64
3 | Updated       | subscription-manager-gui-1.10.11-1.el6.x86_64
3 | Update        | 1.10.17-1.el6_1.x86_64
1 | Install       | subscription-manager-1.10.11-1.el6.x86_64
1 | Install       | subscription-manager-firstboot-1.10.11-1.el6.x86_64
1 | Install       | subscription-manager-gui-1.10.11-1.el6.x86_64
history package-list
```

In this example, three packages were installed during the initial system installation: *subscription-manager*, *subscription-manager-firstboot*, and *subscription-manager-gui*. In the third transaction, all these packages were updated from version 0.95.11 to version 0.95.17.

5.4.2. Examining Transactions

To display the summary of a single transaction, as **root**, use the **yum history summary** command in the following form:

```
yum history summary id
```

To examine a particular transaction or transactions in more detail, run the following command as **root**:

```
yum history info id...
```

The *id* argument is optional and when you omit it, **yum** automatically uses the last transaction. Note that when specifying more than one transaction, you can also use a range:

```
yum history info start_id..end_id
```

Example 5.22. Example output of yum history info

The following is sample output for two transactions, each installing one new package:

```
~]# yum history info 4..5
Loaded plugins: langpacks, product-id, subscription-manager
Transaction ID : 4..5
Begin time      : Thu Jul 21 15:10:46 2011
Begin rpmdb     : 1107:0c67c32219c199f92ed8da7572b4c6df64eacd3a
End time        : 15:33:15 2011 (22 minutes)
End rpmdb       : 1109:1171025bd9b6b5f8db30d063598f590f1c1f3242
User           : Jaromir Hradilek <jhradilek>
Return-Code     : Success
Command Line    : install screen
Command Line    : install yum-plugin-aliases
Transaction performed with:
  Installed      rpm-4.8.0-16.el6.x86_64
  Installed      yum-3.2.29-17.el6.noarch
  Installed      yum-metadata-parser-1.1.2-16.el6.x86_64
Packages Altered:
  Install screen-4.0.3-16.el6.x86_64
  Install yum-plugin-aliases-1.1.30-6.el6.noarch
history info
```

You can also view additional information, such as what configuration options were used at the time of the transaction, or from what repository and why were certain packages installed. To determine what additional information is available for a certain transaction, type the following at a shell prompt as **root**:

```
yum history addon-info id
```

Similarly to **yum history info**, when no *id* is provided, **yum** automatically uses the latest transaction. Another way to refer to the latest transaction is to use the **last** keyword:

```
yum history addon-info last
```

Example 5.23. Example output of yum history addon-info

For the fourth transaction in the history, the **yum history addon-info** command provides the following output:

```
~]# yum history addon-info 4
Loaded plugins: langpacks, product-id, subscription-manager
Transaction ID: 4
Available additional history information:
  config-main
  config-repos
  saved_tx

history addon-info
```

In the output of the **yum history addon-info** command, three types of information are available:

- ▶ **config-main** — global Yum options that were in use during the transaction. See [Section 5.5.1, “Setting \[main\] Options”](#) for information on how to change global options.

- ▶ **config-repos** — options for individual Yum repositories. See [Section 5.5.2, “Setting \[repository\] Options”](#) for information on how to change options for individual repositories.
- ▶ **saved_tx** — the data that can be used by the **yum load-transaction** command in order to repeat the transaction on another machine (see below).

To display selected type of additional information, run the following command as **root**:

```
yum history addon-info id information
```

5.4.3. Reverting and Repeating Transactions

Apart from reviewing the transaction history, the **yum history** command provides means to revert or repeat a selected transaction. To revert a transaction, type the following at a shell prompt as **root**:

```
yum history undo id
```

To repeat a particular transaction, as **root**, run the following command:

```
yum history redo id
```

Both commands also accept the **last** keyword to undo or repeat the latest transaction.

Note that both **yum history undo** and **yum history redo** commands only revert or repeat the steps that were performed during a transaction. If the transaction installed a new package, the **yum history undo** command will uninstall it, and if the transaction uninstalled a package the command will again install it. This command also attempts to downgrade all updated packages to their previous version, if these older packages are still available.

When managing several identical systems, Yum also allows you to perform a transaction on one of them, store the transaction details in a file, and after a period of testing, repeat the same transaction on the remaining systems as well. To store the transaction details to a file, type the following at a shell prompt as **root**:

```
yum -q history addon-info id saved_tx > file_name
```

Once you copy this file to the target system, you can repeat the transaction by using the following command as **root**:

```
yum load-transaction file_name
```

You can configure **load-transaction** to ignore missing packages or rpmbdb version. For more information on these configuration options see the `yum.conf` man page.

5.4.4. Starting New Transaction History

Yum stores the transaction history in a single SQLite database file. To start new transaction history, run the following command as **root**:

```
yum history new
```

This will create a new, empty database file in the `/var/lib/yum/history/` directory. The old transaction history will be kept, but will not be accessible as long as a newer database file is present in the directory.

5.5. Configuring Yum and Yum Repositories

The configuration file for `yum` and related utilities is located at `/etc/yum.conf`. This file contains one mandatory `[main]` section, which allows you to set Yum options that have global effect, and can also contain one or more `[repository]` sections, which allow you to set repository-specific options. However, it is recommended to define individual repositories in new or existing `.repo` files in the `/etc/yum.repos.d/` directory. The values you define in the `[main]` section of the `/etc/yum.conf` file can override values set in individual `[repository]` sections.

This section shows you how to:

- ▶ set global Yum options by editing the `[main]` section of the `/etc/yum.conf` configuration file;
- ▶ set options for individual repositories by editing the `[repository]` sections in `/etc/yum.conf` and `.repo` files in the `/etc/yum.repos.d/` directory;
- ▶ use Yum variables in `/etc/yum.conf` and files in the `/etc/yum.repos.d/` directory so that dynamic version and architecture values are handled correctly;
- ▶ add, enable, and disable Yum repositories on the command line; and,
- ▶ set up your own custom Yum repository.

5.5.1. Setting [main] Options

The `/etc/yum.conf` configuration file contains exactly one `[main]` section, and while some of the key-value pairs in this section affect how `yum` operates, others affect how Yum treats repositories. You can add many additional options under the `[main]` section heading in `/etc/yum.conf`.

A sample `/etc/yum.conf` configuration file can look like this:

```
[main]
cachedir=/var/cache/yum/$basearch/$releasew
keepcache=0
debuglevel=2
logfile=/var/log/yum.log
exactarch=1
obsoletes=1
gpgcheck=1
plugins=1
installonly_limit=3

[comments abridged]

# PUT YOUR REPOS HERE OR IN separate files named file.repo
# in /etc/yum.repos.d
```

The following are the most commonly-used options in the `[main]` section:

assumeyes=value

The **assumeyes** option determines whether or not yum prompts for confirmation of critical actions. Replace *value* with one of:

0 — (*default*). yum should prompt for confirmation of critical actions it performs.

1 — Do not prompt for confirmation of critical yum actions. If **assumeyes=1** is set, yum behaves in the same way as the command line options **-y** and **--assumeyes**.

cachedir=directory

Use this option to set the directory where Yum should store its cache and database files. Replace *directory* with an absolute path to the directory. By default, Yum's cache directory is **/var/cache/yum/\$basearch/\$releasever**.

See [Section 5.5.3, “Using Yum Variables”](#) for descriptions of the **\$basearch** and **\$releasever** Yum variables.

debuglevel=value

This option specifies the detail of debugging output produced by yum. Here, *value* is an integer between **1** and **10**. Setting a higher **debuglevel** value causes yum to display more detailed debugging output. **debuglevel=0** disables debugging output, while **debuglevel=2** is the default.

exactarch=value

With this option, you can set yum to consider exact architecture when updating already installed packages. Replace *value* with:

0 — Do not take into account the exact architecture when updating packages.

1 — (*default*). Consider the exact architecture when updating packages. With this setting, yum does not install an i686 package to update an i386 package already installed on the system.

exclude=package_name [more_package_names]

The **exclude** option allows you to exclude packages by keyword during installation or updating. Listing multiple packages for exclusion can be accomplished by quoting a space-delimited list of packages. Shell glob expressions using wildcards (for example, ***** and **?**) are allowed.

gpgcheck=value

Use the **gpgcheck** option to specify if yum should perform a GPG signature check on packages. Replace *value* with:

0 — Disable GPG signature-checking on packages in all repositories, including local package installation.

1 — (*default*). Enable GPG signature-checking on all packages in all repositories, including local package installation. With **gpgcheck** enabled, all packages' signatures are checked.

If this option is set in the **[main]** section of the **/etc/yum.conf** file, it sets the GPG-checking rule for all repositories. However, you can also set **gpgcheck=value** for individual repositories instead; that is, you can enable GPG-checking on one repository while disabling it on another. Setting **gpgcheck=value** for an individual repository in its corresponding **.repo** file overrides the default if it is present in **/etc/yum.conf**.

For more information on GPG signature-checking, see [Section A.3.2, “Checking Package Signatures”](#).

group_command=value

Use the **group_command** option to specify how the **yum group install**, **yum group upgrade**, and **yum group remove** commands handle a package group. Replace *value* with:

simple — Install all members of a package group. Upgrade only previously installed packages, but do not install packages that were added to the group in the meantime.

compat — Similar to **simple** but **yum upgrade** also installs packages that were added to the group since the previous upgrade.

objects — (*default*.) With this option, yum keeps track of the previously-installed groups and distinguishes between packages installed as a part of the group and packages installed separately. See [Example 5.14, “Viewing information on the LibreOffice package group”](#).

group_package_types=package_type [more_package_types]

Here you can specify which type of packages (*optional*, *default* or *mandatory*) is installed when the **yum group install** command is called. The *default* and *mandatory* package types are chosen by default.

history_record=value

With this option, you can set **yum** to record transaction history. Replace *value* with one of:

0 — **yum** should *not* record history entries for transactions.

1 — (*default*). **yum** should record history entries for transactions. This operation takes certain amount of disk space, and some extra time in the transactions, but it provides a lot of information about past operations, which can be displayed with the **yum history** command.

history_record=1 is the default.

For more information on the **yum history** command, see [Section 5.4, “Working with Transaction History”](#).



Note

yum uses history records to detect modifications to the **rpmdb** that have been done outside of **yum**. In such case, **yum** displays a warning and automatically searches for possible problems caused by altering **rpmdb**. With **history_record** turned off, **yum** is not able to detect these changes and no automatic checks are performed.

installonlypkgs=space separated list of packages

Here you can provide a space-separated list of packages which **yum** can *install*, but will never *update*. See the **yum.conf(5)** manual page for the list of packages which are install-only by default.

If you add the **installonlypkgs** directive to **/etc/yum.conf**, you should ensure that you list *all* of the packages that should be install-only, including any of those listed under the **installonlypkgs** section of **yum.conf(5)**. In particular, kernel packages should always be listed in **installonlypkgs** (as they are by default), and **installonly_limit** should always

be set to a value greater than **2** so that a backup kernel is always available in case the default one fails to boot.

installonly_limit=value

This option sets how many packages listed in the **installonlypkgs** directive can be installed at the same time. Replace *value* with an integer representing the maximum number of versions that can be installed simultaneously for any single package listed in **installonlypkgs**.

The defaults for the **installonlypkgs** directive include several different kernel packages, so be aware that changing the value of **installonly_limit** also affects the maximum number of installed versions of any single kernel package. The default value listed in **/etc/yum.conf** is **installonly_limit=3**, and it is not recommended to decrease this value, particularly below **2**.

keepcache=value

The **keepcache** option determines whether Yum keeps the cache of headers and packages after successful installation. Here, *value* is one of:

- 0** — (*default*). Do not retain the cache of headers and packages after a successful installation.
- 1** — Retain the cache after a successful installation.

logfile=file_name

To specify the location for logging output, replace *file_name* with an absolute path to the file in which **yum** should write its logging output. By default, **yum** logs to **/var/log/yum.log**.

max_connections=number

Here *value* stands for the maximum number of simultaneous connections, default is 5.

multilib_policy=value

The **multilib_policy** option sets the installation behavior if several architecture versions are available for package install. Here, *value* stands for:

- best** — install the best-choice architecture for this system. For example, setting **multilib_policy=best** on an AMD64 system causes **yum** to install 64-bit versions of all packages.
- all** — always install every possible architecture for every package. For example, with **multilib_policy** set to **all** on an AMD64 system, **yum** would install both the i686 and AMD64 versions of a package, if both were available.

obsoletes=value

The **obsoletes** option enables the obsoletes process logic during updates. When one package declares in its spec file that it *obsoletes* another package, the latter package is replaced by the former package when the former package is installed. Obsoletes are declared, for example, when a package is renamed. Replace *value* with one of:

- 0** — Disable **yum**'s obsoletes processing logic when performing updates.
- 1** — (*default*). Enable **yum**'s obsoletes processing logic when performing updates.

plugins=value

This is a global switch to enable or disable **yum** plug-ins, *value* is one of:

- 0** — Disable all Yum plug-ins globally.



Important

Disabling all plug-ins is not advised because certain plug-ins provide important **Yum** services. In particular, **product-id** and **subscription-manager** plug-ins provide support for the certificate-based **Content Delivery Network** (CDN). Disabling plug-ins globally is provided as a convenience option, and is generally only recommended when diagnosing a potential problem with **Yum**.

1 — (*default*). Enable all Yum plug-ins globally. With **plugins=1**, you can still disable a specific Yum plug-in by setting **enabled=0** in that plug-in's configuration file.

For more information about various Yum plug-ins, see [Section 5.6, “Yum Plug-ins”](#). For further information on controlling plug-ins, see [Section 5.6.1, “Enabling, Configuring, and Disabling Yum Plug-ins”](#).

reposdir=directory

Here, *directory* is an absolute path to the directory where **.repo** files are located. All **.repo** files contain repository information (similar to the **[repository]** sections of **/etc/yum.conf**). **yum** collects all repository information from **.repo** files and the **[repository]** section of the **/etc/yum.conf** file to create a master list of repositories to use for transactions. If **reposdir** is not set, **yum** uses the default directory **/etc/yum.repos.d/**.

retries=value

This option sets the number of times **yum** should attempt to retrieve a file before returning an error. *value* is an integer **0** or greater. Setting value to **0** makes **yum** retry forever. The default value is **10**.

For a complete list of available **[main]** options, see the **[main] OPTIONS** section of the **yum.conf(5)** manual page.

5.5.2. Setting **[repository]** Options

The **[repository]** sections, where *repository* is a unique repository ID such as **my_personal_repo** (spaces are not permitted), allow you to define individual Yum repositories.

The following is a bare-minimum example of the form a **[repository]** section takes:

```
[repository]
name=repository_name
baseurl=repository_url
```

Every **[repository]** section must contain the following directives:

name=repository_name

Here, *repository_name* is a human-readable string describing the repository.

baseurl=repository_url

Replace *repository_url* with a URL to the directory where the repodata directory of a repository is located:

- ▶ If the repository is available over HTTP, use: **http://path/to/repo**
- ▶ If the repository is available over FTP, use: **ftp://path/to/repo**
- ▶ If the repository is local to the machine, use: **file:///path/to/local/repo**
- ▶ If a specific online repository requires basic HTTP authentication, you can specify your username and password by prepending it to the URL as **username:password@link**. For example, if a repository on `http://www.example.com/repo/` requires a username of “user” and a password of “password”, then the **baseurl** link could be specified as `http://user:password@www.example.com/repo/`.

Usually this URL is an HTTP link, such as:

```
baseurl=http://path/to/repo/releases/$releasever/server/$basearch/os/
```

Note that Yum always expands the **\$releasever**, **\$arch**, and **\$basearch** variables in URLs. For more information about Yum variables, see [Section 5.5.3, “Using Yum Variables”](#).

Other useful [**repository**] directive are:

enabled=value

This is a simple way to tell **yum** to use or ignore a particular repository, *value* is one of:

0 — Do not include this repository as a package source when performing updates and installs.
This is an easy way of quickly turning repositories on and off, which is useful when you desire a single package from a repository that you do not want to enable for updates or installs.

1 — Include this repository as a package source.

Turning repositories on and off can also be performed by passing either the `--enablerrepo=repo_name` or `--disablerrepo=repo_name` option to **yum**, or through the **Add/Remove Software** window of the **PackageKit** utility.

async=value

Controls parallel downloading of repository packages. Here, *value* is one of:

auto — (*default*) parallel downloading is used if possible, which means that **yum** automatically disables it for repositories created by plug-ins to avoid failures.

on — parallel downloading is enabled the repository.

off — parallel downloading is disabled the repository.

Many more [**repository**] options exist, part of them has the same form and function as certain [**main**] options. For a complete list, see the [**repository**] **OPTIONS** section of the **yum.conf(5)** manual page.

Example 5.24. A sample /etc/yum.repos.d/redhat.repo file

The following is a sample `/etc/yum.repos.d/redhat.repo` file:

```

#
# Red Hat Repositories
# Managed by (rhsm) subscription-manager
#

[red-hat-enterprise-linux-scalable-file-system-for-rhel-6-entitlement-rpms]
name = Red Hat Enterprise Linux Scalable File System (for RHEL 6 Entitlement)
(RPMs)
baseurl = https://cdn.redhat.com/content/dist/rhel/entitlement-
6/releases/$releasever/$basearch/scalablefilesystem/os
enabled = 1
gpgcheck = 1
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
sslverify = 1
sslcacert = /etc/rhsm/ca/redhat-uep.pem
sslclientkey = /etc/pki/entitlement/key.pem
sslclientcert = /etc/pki/entitlement/11300387955690106.pem

[red-hat-enterprise-linux-scalable-file-system-for-rhel-6-entitlement-source-
rpms]
name = Red Hat Enterprise Linux Scalable File System (for RHEL 6 Entitlement)
(Source RPMs)
baseurl = https://cdn.redhat.com/content/dist/rhel/entitlement-
6/releases/$releasever/$basearch/scalablefilesystem/source/SRPMS
enabled = 0
gpgcheck = 1
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
sslverify = 1
sslcacert = /etc/rhsm/ca/redhat-uep.pem
sslclientkey = /etc/pki/entitlement/key.pem
sslclientcert = /etc/pki/entitlement/11300387955690106.pem

[red-hat-enterprise-linux-scalable-file-system-for-rhel-6-entitlement-debug-rpms]
name = Red Hat Enterprise Linux Scalable File System (for RHEL 6 Entitlement)
(Debug RPMs)
baseurl = https://cdn.redhat.com/content/dist/rhel/entitlement-
6/releases/$releasever/$basearch/scalablefilesystem/debug
enabled = 0
gpgcheck = 1
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
sslverify = 1
sslcacert = /etc/rhsm/ca/redhat-uep.pem
sslclientkey = /etc/pki/entitlement/key.pem
sslclientcert = /etc/pki/entitlement/11300387955690106.pem

```

5.5.3. Using Yum Variables

You can use and reference the following built-in variables in `yum` commands and in all Yum configuration files (that is, `/etc/yum.conf` and all `.repo` files in the `/etc/yum.repos.d/` directory):

\$releasever

You can use this variable to reference the release version of Red Hat Enterprise Linux. Yum obtains the value of `$releasever` from the `distroverpkg=value` line in the `/etc/yum.conf` configuration file. If there is no such line in `/etc/yum.conf`, then yum infers the correct value by

deriving the version number from the *redhat-release* package.

\$arch

You can use this variable to refer to the system's CPU architecture as returned when calling Python's `os.uname()` function. Valid values for `$arch` include: `i586`, `i686` and `x86_64`.

\$basearch

You can use `$basearch` to reference the base architecture of the system. For example, i686 and i586 machines both have a base architecture of `i386`, and AMD64 and Intel64 machines have a base architecture of `x86_64`.

\$YUM0-9

These ten variables are each replaced with the value of any shell environment variables with the same name. If one of these variables is referenced (in `/etc/yum.conf` for example) and a shell environment variable with the same name does not exist, then the configuration file variable is not replaced.

To define a custom variable or to override the value of an existing one, create a file with the same name as the variable (without the "\$" sign) in the `/etc/yum/vars/` directory, and add the desired value on its first line.

For example, repository descriptions often include the operating system name. To define a new variable called `$osname`, create a new file with "Red Hat Enterprise Linux" on the first line and save it as `/etc/yum/vars/osname`:

```
~]# echo "Red Hat Enterprise Linux 7" > /etc/yum/vars/osname
```

Instead of "Red Hat Enterprise Linux 7", you can now use the following in the `.repo` files:

```
name=$osname $releasever
```

Note

Creating custom variables is currently impacted by an issue with variables substitution ([BZ#1102585](#)).

5.5.4. Viewing the Current Configuration

To display the current values of global Yum options (that is, the options specified in the `[main]` section of the `/etc/yum.conf` file), run the `yum-config-manager` with no command line options:

```
yum-config-manager
```

To list the content of a different configuration section or sections, use the command in the following form:

```
yum-config-manager section...
```

You can also use a glob expression to display the configuration of all matching sections:

```
yum-config-manager glob_expression...
```

Example 5.25. Viewing configuration of the main section

To list all configuration options and their corresponding values for the main section, type the following at a shell prompt:

```
~]$ yum-config-manager main \*
Loaded plugins: langpacks, product-id, subscription-manager
=====
[main]
alwaysprompt = True
assumeyes = False
bandwidth = 0
bugtracker_url = https://bugzilla.redhat.com/enter_bug.cgi?
product=Red%20Hat%20Enterprise%20Linux%206&component=yum
cache = 0
[output truncated]
```

5.5.5. Adding, Enabling, and Disabling a Yum Repository

[Section 5.5.2, “Setting \[repository\] Options”](#) described various options you can use to define a Yum repository. This section explains how to add, enable, and disable a repository by using the **yum-config-manager** command.



Important

When the system is registered with Red Hat Subscription Management to the certificate-based **Content Delivery Network** (CDN), the **Red Hat Subscription Manager** tools are used to manage repositories in the `/etc/yum.repos.d/redhat.repo` file.

Adding a Yum Repository

To define a new repository, you can either add a **[repository]** section to the `/etc/yum.conf` file, or to a `.repo` file in the `/etc/yum.repos.d/` directory. All files with the `.repo` file extension in this directory are read by **yum**, and it is recommended to define your repositories here instead of in `/etc/yum.conf`.



Warning

Obtaining and installing software packages from unverified or untrusted software sources other than Red Hat’s certificate-based **Content Delivery Network** (CDN) constitutes a potential security risk, and could lead to security, stability, compatibility, and maintainability issues.

Yum repositories commonly provide their own `.repo` file. To add such a repository to your system and enable it, run the following command as **root**:

```
yum-config-manager --add-repo repository_url
```

...where `repository_url` is a link to the `.repo` file.

Example 5.26. Adding example.repo

To add a repository located at <http://www.example.com/example.repo>, type the following at a shell prompt:

```
~]# yum-config-manager --add-repo http://www.example.com/example.repo
Loaded plugins: langpacks, product-id, subscription-manager
adding repo from: http://www.example.com/example.repo
grabbing file http://www.example.com/example.repo to
/etc/yum.repos.d/example.repo
example.repo | 413 B      00:00
repo saved to /etc/yum.repos.d/example.repo
```

Enabling a Yum Repository

To enable a particular repository or repositories, type the following at a shell prompt as **root**:

```
yum-config-manager --enable repository...
```

...where *repository* is the unique repository ID (use **yum repolist all** to list available repository IDs). Alternatively, you can use a glob expression to enable all matching repositories:

```
yum-config-manager --enable glob_expression...
```

Example 5.27. Enabling repositories defined in custom sections of /etc/yum.conf.

To enable repositories defined in the **[example]**, **[example-debuginfo]**, and **[example-source]** sections, type:

```
~]# yum-config-manager --enable example\*
Loaded plugins: langpacks, product-id, subscription-manager
=====
repo: example =====
[example]
bandwidth = 0
base_persistdir = /var/lib/yum/repos/x86_64/6Server
baseurl = http://www.example.com/repo/6Server/x86_64/
cache = 0
cachedir = /var/cache/yum/x86_64/6Server/example
[output truncated]
```

When successful, the **yum-config-manager --enable** command displays the current repository configuration.

Disabling a Yum Repository

To disable a Yum repository, run the following command as **root**:

```
yum-config-manager --disable repository...
```

...where *repository* is the unique repository ID (use **yum repolist all** to list available repository IDs). Similarly to **yum-config-manager --enable**, you can use a glob expression to disable all matching repositories at the same time:

```
yum-config-manager --disable glob_expression...
```

When successful, the **yum-config-manager --disable** command displays the current configuration.

5.5.6. Creating a Yum Repository

To set up a Yum repository, follow these steps:

1. Install the *createrepo* package. To do so, type the following at a shell prompt as **root**:

```
yum install createrepo
```

2. Copy all packages that you want to have in your repository into one directory, such as **/mnt/local_repo/**.

3. Change to this directory and run the following command:

```
createrepo --database /mnt/local_repo
```

This creates the necessary metadata for your Yum repository, as well as the **sqlite** database for speeding up **yum** operations.

5.6. Yum Plug-ins

Yum provides plug-ins that extend and enhance its operations. Certain plug-ins are installed by default. Yum always informs you which plug-ins, if any, are loaded and active whenever you call any **yum** command. For example:

```
[~]# yum info yum
Loaded plugins: langpacks, product-id, subscription-manager
[output truncated]
```

Note that the plug-in names which follow **Loaded plugins** are the names you can provide to the **--disableplugins=plugin_name** option.

5.6.1. Enabling, Configuring, and Disabling Yum Plug-ins

To enable Yum plug-ins, ensure that a line beginning with **plugins=** is present in the **[main]** section of **/etc/yum.conf**, and that its value is **1**:

```
plugins=1
```

You can disable all plug-ins by changing this line to **plugins=0**.



Important

Disabling all plug-ins is not advised because certain plug-ins provide important Yum services. In particular, **product-id** and **subscription-manager** plug-ins provide support for the certificate-based **Content Delivery Network** (CDN). Disabling plug-ins globally is provided as a convenience option, and is generally only recommended when diagnosing a potential problem with Yum.

Every installed plug-in has its own configuration file in the `/etc/yum/pluginconf.d/` directory. You can set plug-in specific options in these files. For example, here is the **aliases** plug-in's **aliases.conf** configuration file:

```
[main]
enabled=1
```

Plug-in configuration files always contain a **[main]** section (similar to Yum's `/etc/yum.conf` file) in which there is (or you can place if it is missing) an **enabled=** option that controls whether the plug-in is enabled when you run **yum** commands.

If you disable all plug-ins by setting **enabled=0** in `/etc/yum.conf`, then all plug-ins are disabled regardless of whether they are enabled in their individual configuration files.

If you merely want to disable all Yum plug-ins for a single **yum** command, use the **--nopugins** option.

If you want to disable one or more Yum plug-ins for a single **yum** command, add the **--disableplugin=plugin_name** option to the command. For example, to disable the **aliases** plug-in while updating a system, type:

```
~]# yum update --disableplugin=aliases
```

The plug-in names you provide to the **--disableplugin=** option are the same names listed after the **Loaded plugins** line in the output of any **yum** command. You can disable multiple plug-ins by separating their names with commas. In addition, you can match multiple plug-in names or shorten long ones by using glob expressions:

```
~]# yum update --disableplugin=aliases, lang*
```

5.6.2. Installing Additional Yum Plug-ins

Yum plug-ins usually adhere to the `yum-plugin-plugin_name` package-naming convention, but not always: the package which provides the **kabi** plug-in is named **kabi-yum-plugins**, for example. You can install a Yum plug-in in the same way you install other packages. For instance, to install the **yum-aliases** plug-in, type the following at a shell prompt:

```
~]# yum install yum-plugin-aliases
```

5.6.3. Working with Yum Plug-ins

The following list provides descriptions and usage instructions for several useful Yum plug-ins. Plug-ins are listed by names, brackets contain the name of package.

kabi (kabi-yum-plugins)

The **kabi** plug-in checks whether a driver update package conforms with official Red Hat *kernel Application Binary Interface* (kABI). With this plug-in enabled, when a user attempts to install a package that uses kernel symbols which are not on a whitelist, a warning message is written to the system log. Additionally, configuring the plug-in to run in enforcing mode prevents such packages from being installed at all.

To configure the **kabi** plug-in, edit the configuration file located in `/etc/yum/pluginconf.d/kabi.conf`. A list of directives that can be used in the `[main]` section is shown in the following table.

Table 5.3. Supported kabi.conf directives

Directive	Description
<code>enabled=value</code>	Allows you to enable or disable the plug-in. The <i>value</i> must be either 1 (enabled), or 0 (disabled). When installed, the plug-in is enabled by default.
<code>whitelists=directory</code>	Allows you to specify the <i>directory</i> in which the files with supported kernel symbols are located. By default, the kabi plug-in uses files provided by the <i>kernel-abi-whitelists</i> package (that is, the <code>/lib/modules/kabi-rhel70/</code> directory).
<code>enforce=value</code>	Allows you to enable or disable enforcing mode. The <i>value</i> must be either 1 (enabled), or 0 (disabled). By default, this option is commented out and the kabi plug-in only displays a warning message.

product-id (subscription-manager)

The **product-id** plug-in manages product identity certificates for products installed from the Content Delivery Network. The **product-id** plug-in is installed by default.

langpacks (yum-langpacks)

The **langpacks** plug-in is used to search for locale packages of a selected language for every package that is installed. The **langpacks** plug-in is installed by default.

aliases (yum-plugin-aliases)

The **aliases** plug-in adds the **alias** command-line option which allows to configure and use aliases for yum commands.

yum-changelog (yum-plugin-changelog)

The **yum-changelog** plug-in adds the **--changelog** command-line option that allows for viewing package change logs before and after updating.

yum-tmprepo (yum-plugin-tmprepo)

The **yum-tmprepo** plug-in adds the **--tmprepo** command-line option that takes the URL of a repository file, downloads and enables it for only one transaction. This plug-in tries to ensure the safe temporary usage of repositories, by default, it does not allow to disable the gpg check.

yum-verify (yum-plugin-verify)

The **yum-verify** plug-in adds the **verify**, **verify-rpm**, and **verify-all** command-line options for viewing verification data on the system.

yum-versionlock (yum-plugin-versionlock)

The **yum-versionlock** plug-in excludes other versions of selected packages, which allows for protecting packages from being updated by newer versions. With the **versionlock** command-line option, you can view and edit the list of locked packages.

5.7. Additional Resources

For more information on how to manage software packages on Red Hat Enterprise Linux, see the resources listed below.

Installed Documentation

- ▶ **yum(8)** — The manual page for the **yum** command line utility provides a complete list of supported options and commands.
- ▶ **yumdb(8)** — The manual page for the **yumdb** command line utility documents how to use this tool to query and, if necessary, alter the Yum database.
- ▶ **yum .conf(5)** — The manual page named **yum .conf** documents available Yum configuration options.
- ▶ **yum -utils(1)** — The manual page named **yum -utils** lists and briefly describes additional utilities for managing Yum configuration, manipulating repositories, and working with Yum database.

Online Documentation

- ▶ [Yum Guides](#) — The *Yum Guides* page on the project home page provides links to further documentation.

See Also

- ▶ [Chapter 4, Gaining Privileges](#) documents how to gain administrative privileges by using the **su** and **sudo** commands.
- ▶ [Appendix A, RPM](#) describes the **RPM Package Manager** (RPM), the packaging system used by Red Hat Enterprise Linux.

Part III. Infrastructure Services

This part provides information on how to configure services and daemons and enable remote access to a Red Hat Enterprise Linux machine.

Chapter 6. Managing Services with systemd

6.1. Introduction to systemd

Systemd is a system and service manager for Linux operating systems. It is designed to be backwards compatible with SysV init scripts, and provides a number of features such as parallel startup of system services at boot time, on-demand activation of daemons, support for system state snapshots, or dependency-based service control logic. In Red Hat Enterprise Linux 7, systemd replaces Upstart as the default init system.

Systemd introduces the concept of *systemd units*. These units are represented by unit configuration files located in one of the directories listed in [Table 6.2, “Systemd Unit Locations”](#), and encapsulate information about system services, listening sockets, saved system state snapshots, and other objects that are relevant to the init system. For a complete list of available systemd unit types, see [Table 6.1, “Available systemd Unit Types”](#).

Table 6.1. Available systemd Unit Types

Unit Type	File Extension	Description
Service unit	.service	A system service.
Target unit	.target	A group of systemd units.
Automount unit	.automount	A file system automount point.
Device unit	.device	A device file recognized by the kernel.
Mount unit	.mount	A file system mount point.
Path unit	.path	A file or directory in a file system.
Scope unit	.scope	An externally created process.
Slice unit	.slice	A group of hierarchically organized units that manage system processes.
Snapshot unit	.snapshot	A saved state of the systemd manager.
Socket unit	.socket	An inter-process communication socket.
Swap unit	.swap	A swap device or a swap file.
Timer unit	.timer	A systemd timer.

Table 6.2. Systemd Unit Locations

Directory	Description
/usr/lib/systemd/system/	Systemd units distributed with installed RPM packages.
/run/systemd/system/	Systemd units created at run time. This directory takes precedence over the directory with installed service units.
/etc/systemd/system/	Systemd units created and managed by the system administrator. This directory takes precedence over the directory with runtime units.

6.1.1. Main Features

In Red Hat Enterprise Linux 7, the systemd system and service manager provides the following main features:

- ▶ **Socket-based activation** — At boot time, systemd creates listening sockets for all system services that support this type of activation, and passes the sockets to these services as soon as they are started. This not only allows systemd to start services in parallel, but also makes it possible to restart a service without losing any message sent to it while it is unavailable: the corresponding socket remains accessible and all messages are queued.

Systemd uses *socket units* for socket-based activation.

- ▶ **Bus-based activation** — System services that use D-Bus for inter-process communication can be started on-demand the first time a client application attempts to communicate with them. Systemd uses *D-Bus service files* for bus-based activation.
- ▶ **Device-based activation** — System services that support device-based activation can be started on-demand when a particular type of hardware is plugged in or becomes available. Systemd uses *device units* for device-based activation.
- ▶ **Path-based activation** — System services that support path-based activation can be started on-demand when a particular file or directory changes its state. Systemd uses *path units* for path-based activation.
- ▶ **System state snapshots** — Systemd can temporarily save the current state of all units or restore a previous state of the system from a dynamically created snapshot. To store the current state of the system, systemd uses dynamically created *snapshot units*.
- ▶ **Mount and automount point management** — Systemd monitors and manages mount and automount points. Systemd uses *mount units* for mount points and *automount units* for automount points.
- ▶ **Aggressive parallelization** — Because of the use of socket-based activation, systemd can start system services in parallel as soon as all listening sockets are in place. In combination with system services that support on-demand activation, parallel activation significantly reduces the time required to boot the system.
- ▶ **Transactional unit activation logic** — Before activating or deactivating a unit, systemd calculates its dependencies, creates a temporary transaction, and verifies that this transaction is consistent. If a transaction is inconsistent, systemd automatically attempts to correct it and remove non-essential jobs from it before reporting an error.
- ▶ **Backwards compatibility with SysV init** — Systemd fully supports SysV init scripts as described in the *Linux Standard Base Core Specification*, which eases the upgrade path to systemd service units.

6.1.2. Compatibility Changes

The systemd system and service manager is designed to be mostly compatible with SysV init and Upstart. The following are the most notable compatibility changes with regards to the previous major release of the Red Hat Enterprise Linux system:

- ▶ Systemd has only limited support for runlevels. It provides a number of target units that can be directly mapped to these runlevels and for compatibility reasons, it is also distributed with the earlier **runlevel** command. Not all systemd targets can be directly mapped to runlevels, however, and as a consequence, this command may return **N** to indicate an unknown runlevel. It is recommended that you avoid using the **runlevel** command if possible.

For more information about systemd targets and their comparison with runlevels, see [Section 6.3, “Working with systemd Targets”](#).

- ▶ The `systemctl` utility does not support custom commands. In addition to standard commands such as `start`, `stop`, and `status`, authors of SysV init scripts could implement support for any number of arbitrary commands in order to provide additional functionality. For example, the init script for `iptables` in Red Hat Enterprise Linux 6 could be executed with the `panic` command, which immediately enabled panic mode and reconfigured the system to start dropping all incoming and outgoing packets. This is not supported in `systemd` and the `systemctl` only accepts documented commands.

For more information about the `systemctl` utility and its comparison with the earlier `service` utility, see [Section 6.2, “Managing System Services”](#).

- ▶ The `systemctl` utility does not communicate with services that have not been started by `systemd`. When `systemd` starts a system service, it stores the ID of its main process in order to keep track of it. The `systemctl` utility then uses this PID to query and manage the service. Consequently, if a user starts a particular daemon directly on the command line, `systemctl` is unable to determine its current status or stop it.
- ▶ `Systemd` stops only running services. Previously, when the shutdown sequence was initiated, Red Hat Enterprise Linux 6 and earlier releases of the system used symbolic links located in the `/etc/rc0.d/` directory to stop all available system services regardless of their status. With `systemd`, only running services are stopped on shutdown.
- ▶ System services are unable to read from the standard input stream. When `systemd` starts a service, it connects its standard input to `/dev/null` to prevent any interaction with the user.
- ▶ System services do not inherit any context (such as the `HOME` and `PATH` environment variables) from the invoking user and their session. Each service runs in a clean execution context.
- ▶ `Systemd` reads dependency information encoded in the Linux Standard Base (LSB) header and interprets it at run time.
- ▶ All operations on service units are subject to a timeout of 5 minutes to prevent a malfunctioning service from freezing the system.

For a detailed list of compatibility changes introduced with `systemd`, see the [Migration Planning Guide](#) for Red Hat Enterprise Linux 7.

6.2. Managing System Services

Previous versions of Red Hat Enterprise Linux, which were distributed with SysV init or Upstart, used *init scripts* located in the `/etc/rc.d/init.d/` directory. These init scripts were typically written in Bash, and allowed the system administrator to control the state of services and daemons in their system. In Red Hat Enterprise Linux 7, these init scripts have been replaced with *service units*.

Service units end with the `.service` file extension and serve a similar purpose as init scripts. To view, start, stop, restart, enable, or disable system services, use the `systemctl` command as described in [Table 6.3, “Comparison of the service Utility with systemctl”](#), [Table 6.4, “Comparison of the chkconfig Utility with systemctl”](#), and in the section below. The `service` and `chkconfig` commands are still available in the system and work as expected, but are only included for compatibility reasons and should be avoided.

**Note**

For clarity, all examples in the rest of this section use full unit names with the `.service` file extension, for example:

```
~]# systemctl stop bluetooth.service
```

When working with system services, it is possible to omit this file extension to reduce typing: when the `systemctl` utility encounters a unit name without a file extension, it automatically assumes it is a service unit. The following command is equivalent to the one above:

```
~]# systemctl stop bluetooth
```

Table 6.3. Comparison of the service Utility with systemctl

service	systemctl	Description
<code>service name start</code>	<code>systemctl start name.service</code>	Starts a service.
<code>service name stop</code>	<code>systemctl stop name.service</code>	Stops a service.
<code>service name restart</code>	<code>systemctl restart name.service</code>	Restarts a service.
<code>service name condrestart</code>	<code>systemctl try-restart name.service</code>	Restarts a service only if it is running.
<code>service name reload</code>	<code>systemctl reload name.service</code>	Reloads configuration.
<code>service name status</code>	<code>systemctl status name.service</code> <code>systemctl is-active name.service</code>	Checks if a service is running.
<code>service --status-all</code>	<code>systemctl list-units --type service --all</code>	Displays the status of all services.

Table 6.4. Comparison of the chkconfig Utility with systemctl

chkconfig	systemctl	Description
<code>chkconfig name on</code>	<code>systemctl enable name.service</code>	Enables a service.
<code>chkconfig name off</code>	<code>systemctl disable name.service</code>	Disables a service.
<code>chkconfig --list name</code>	<code>systemctl status name.service</code> <code>systemctl is-enabled name.service</code>	Checks if a service is enabled.
<code>chkconfig --list</code>	<code>systemctl list-unit-files --type service</code>	Lists all services and checks if they are enabled.

6.2.1. Listing Services

To list all currently loaded service units, type the following at a shell prompt:

```
systemctl list-units --type service
```

For each service unit, this command displays its full name (**UNIT**) followed by a note whether the unit has been loaded (**LOAD**), its high-level (**ACTIVE**) and low-level (**SUB**) unit activation state, and a short description (**DESCRIPTION**).

By default, the **systemctl list-units** command displays only active units. If you want to list all loaded units regardless of their state, run this command with the **--all** or **-a** command line option:

```
systemctl list-units --type service --all
```

You can also list all available service units to see if they are enabled. To do so, type:

```
systemctl list-unit-files --type service
```

For each service unit, this command displays its full name (**UNIT FILE**) followed by information whether the service unit is enabled or not (**STATE**). For information on how to determine the status of individual service units, see [Section 6.2.2, “Displaying Service Status”](#).

Example 6.1. Listing Services

To list all currently loaded service units, run the following command:

```
~]$ systemctl list-units --type service
UNIT                      LOAD ACTIVE SUB DESCRIPTION
abrt-ccpp.service          loaded active exited  Install ABRT coredump hook
abrt-oops.service          loaded active running ABRT kernel log watcher
abrt-vmcore.service        loaded active exited  Harvest vmcores for ABRT
abrt-xorg.service          loaded active running ABRT Xorg log watcher
abrtd.service               loaded active running ABRT Automated Bug Reporting
Tool
...
systemd-vconsole-setup.service loaded active exited  Setup Virtual Console
tog-pegasus.service         loaded active running OpenPegasus CIM Server

LOAD  = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.

46 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'
```

To list all installed service units to determine if they are enabled, type:

```
~]$ systemctl list-unit-files --type service
UNIT FILE                      STATE
abrt-ccpp.service               enabled
abrt-oops.service                enabled
abrt-vmcore.service              enabled
abrt-xorg.service                enabled
abrtd.service                     enabled
...
wpa_supplicant.service           disabled
ypbind.service                    disabled

208 unit files listed.
```

6.2.2. Displaying Service Status

To display detailed information about a service unit that corresponds to a system service, type the following at a shell prompt:

```
systemctl status name.service
```

Replace *name* with the name of the service unit you want to inspect (for example, **gdm**). This command displays the name of the selected service unit followed by its short description, one or more fields described in [Table 6.5, “Available Service Unit Information”](#), and if it is executed by the **root** user, also the most recent log entries.

Table 6.5. Available Service Unit Information

Field	Description
Loaded	Information whether the service unit has been loaded, the absolute path to the unit file, and a note whether the unit is enabled.
Active	Information whether the service unit is running followed by a time stamp.
Main PID	The PID of the corresponding system service followed by its name.
Status	Additional information about the corresponding system service.
Process	Additional information about related processes.
CGroup	Additional information about related Control Groups.

To only verify that a particular service unit is running, run the following command:

```
systemctl is-active name.service
```

Similarly, to determine whether a particular service unit is enabled, type:

```
systemctl is-enabled name.service
```

Note that both **systemctl is-active** and **systemctl is-enabled** return an exit status of **0** if the specified service unit is running or enabled. For information on how to list all currently loaded service units, see [Section 6.2.1, “Listing Services”](#).

Example 6.2. Displaying Service Status

The service unit for the GNOME Display Manager is named **gdm.service**. To determine the current status of this service unit, type the following at a shell prompt:

```
~]# systemctl status gdm.service
gdm.service - GNOME Display Manager
   Loaded: loaded (/usr/lib/systemd/system/gdm.service; enabled)
     Active: active (running) since Thu 2013-10-17 17:31:23 CEST; 5min ago
       Main PID: 1029 (gdm)
          CGroup: /system.slice/gdm.service
                  └─1029 /usr/sbin/gdm
                      ├─1037 /usr/libexec/gdm-simple-slave --display-id /org/gno...
                          └─1047 /usr/bin/Xorg :0 -background none -verbose -auth /r...

Oct 17 17:31:23 localhost systemd[1]: Started GNOME Display Manager.
```

6.2.3. Starting a Service

To start a service unit that corresponds to a system service, type the following at a shell prompt as **root**:

```
systemctl start name.service
```

Replace *name* with the name of the service unit you want to start (for example, **gdm**). This command starts the selected service unit in the current session. For information on how to enable a service unit to be started at boot time, see [Section 6.2.6, “Enabling a Service”](#). For information on how to determine the status of a certain service unit, see [Section 6.2.2, “Displaying Service Status”](#).

Example 6.3. Starting a Service

The service unit for the Apache HTTP Server is named **httpd.service**. To activate this service unit and start the **httpd** daemon in the current session, run the following command as **root**:

```
~]# systemctl start httpd.service
```

6.2.4. Stopping a Service

To stop a service unit that corresponds to a system service, type the following at a shell prompt as **root**:

```
systemctl stop name.service
```

Replace *name* with the name of the service unit you want to stop (for example, **bluetooth**). This command stops the selected service unit in the current session. For information on how to disable a service unit and prevent it from being started at boot time, see [Section 6.2.7, “Disabling a Service”](#). For information on how to determine the status of a certain service unit, see [Section 6.2.2, “Displaying Service Status”](#).

Example 6.4. Stopping a Service

The service unit for the **bluetoothd** daemon is named **bluetooth.service**. To deactivate this service unit and stop the **bluetoothd** daemon in the current session, run the following command as **root**:

```
~]# systemctl stop bluetooth.service
```

6.2.5. Restarting a Service

To restart a service unit that corresponds to a system service, type the following at a shell prompt as **root**:

```
systemctl restart name.service
```

Replace *name* with the name of the service unit you want to restart (for example, **httpd**). This command stops the selected service unit in the current session and immediately starts it again. Importantly, if the selected service unit is not running, this command starts it too. To tell systemd to restart a service unit only if the corresponding service is already running, run the following command as **root**:

```
systemctl try-restart name.service
```

Certain system services also allow you to reload their configuration without interrupting their execution. To do so, type as **root**:

```
systemctl reload name.service
```

Note that system services that do not support this feature ignore this command altogether. For convenience, the **systemctl** command also supports the **reload-or-restart** and **reload-or-try-restart** commands that restart such services instead. For information on how to determine the status of a certain service unit, see [Section 6.2.2, “Displaying Service Status”](#).

Example 6.5. Restarting a Service

In order to prevent users from encountering unnecessary error messages or partially rendered web pages, the Apache HTTP Server allows you to edit and reload its configuration without the need to restart it and interrupt actively processed requests. To do so, type the following at a shell prompt as **root**:

```
~]# systemctl reload httpd.service
```

6.2.6. Enabling a Service

To configure a service unit that corresponds to a system service to be automatically started at boot time, type the following at a shell prompt as **root**:

```
systemctl enable name.service
```

Replace *name* with the name of the service unit you want to enable (for example, **httpd**). This command reads the **[Install]** section of the selected service unit and creates appropriate symbolic links to the **/usr/lib/systemd/system/*name.service*** file in the **/etc/systemd/system/** directory and its subdirectories. This command does not, however, rewrite links that already exist. If you want to ensure that the symbolic links are re-created, use the following command as **root**:

```
systemctl reenable name.service
```

This command disables the selected service unit and immediately enables it again. For information on how to determine whether a certain service unit is enabled to start at boot time, see [Section 6.2.2, “Displaying Service Status”](#). For information on how to start a service in the current session, see [Section 6.2.3, “Starting a Service”](#).

Example 6.6. Enabling a Service

To configure the Apache HTTP Server to start automatically at boot time, run the following command as **root**:

```
~]# systemctl enable httpd.service
ln -s '/usr/lib/systemd/system/httpd.service' '/etc/systemd/system/multi-user.target.wants/httpd.service'
```

6.2.7. Disabling a Service

To prevent a service unit that corresponds to a system service from being automatically started at boot time, type the following at a shell prompt as **root**:

```
systemctl disable name.service
```

Replace *name* with the name of the service unit you want to disable (for example, **bluetooth**). This command reads the **[Install]** section of the selected service unit and removes appropriate symbolic links to the `/usr/lib/systemd/system/name.service` file from the `/etc/systemd/system/` directory and its subdirectories. In addition, you can mask any service unit to prevent it from being started manually or by another service. To do so, run the following command as **root**:

```
systemctl mask name.service
```

This command replaces the `/etc/systemd/system/name.service` file with a symbolic link to `/dev/null`, rendering the actual unit file inaccessible to `systemd`. To revert this action and unmask a service unit, type as **root**:

```
systemctl unmask name.service
```

For information on how to determine whether a certain service unit is enabled to start at boot time, see [Section 6.2.2, “Displaying Service Status”](#). For information on how to stop a service in the current session, see [Section 6.2.4, “Stopping a Service”](#).

Example 6.7. Disabling a Service

[Example 6.4, “Stopping a Service”](#) illustrates how to stop the **bluetooth.service** unit in the current session. To prevent this service unit from starting at boot time, type the following at a shell prompt as **root**:

```
~]# systemctl disable bluetooth.service
rm '/etc/systemd/system/dbus-org.bluez.service'
rm '/etc/systemd/system/bluetooth.target.wants/bluetooth.service'
```

6.3. Working with `systemd` Targets

Previous versions of Red Hat Enterprise Linux, which were distributed with SysV init or Upstart, implemented a predefined set of *runlevels* that represented specific modes of operation. These runlevels were numbered from 0 to 6 and were defined by a selection of system services to be run when a particular runlevel was enabled by the system administrator. In Red Hat Enterprise Linux 7, the concept of runlevels has been replaced with `systemd` *targets*.

`Systemd` targets are represented by *target units*. Target units end with the `.target` file extension and their only purpose is to group together other `systemd` units through a chain of dependencies. For example, the **graphical.target** unit, which is used to start a graphical session, starts system services such as the GNOME Display Manager (**gdm.service**) or Accounts Service (**accounts-daemon.service**) and also activates the **multi-user.target** unit. Similarly, the **multi-user.target** unit starts other essential system services such as NetworkManager (**NetworkManager.service**) or D-Bus (**dbus.service**) and activates another target unit named **basic.target**.

Red Hat Enterprise Linux 7 is distributed with a number of predefined targets that are more or less similar to the standard set of runlevels from the previous releases of this system. For compatibility reasons, it also provides aliases for these targets that directly map them to SysV runlevels. [Table 6.6, “Comparison of SysV Runlevels with systemd Targets”](#) provides a complete list of SysV runlevels and their corresponding systemd targets.

Table 6.6. Comparison of SysV Runlevels with systemd Targets

Runlevel	Target Units	Description
0	<code>runlevel0.target</code> , <code>poweroff.target</code>	Shut down and power off the system.
1	<code>runlevel1.target</code> , <code>rescue.target</code>	Set up a rescue shell.
2	<code>runlevel2.target</code> , <code>multi-user.target</code>	Set up a non-graphical multi-user system.
3	<code>runlevel3.target</code> , <code>multi-user.target</code>	Set up a non-graphical multi-user system.
4	<code>runlevel4.target</code> , <code>multi-user.target</code>	Set up a non-graphical multi-user system.
5	<code>runlevel5.target</code> , <code>graphical.target</code>	Set up a graphical multi-user system.
6	<code>runlevel6.target</code> , <code>reboot.target</code>	Shut down and reboot the system.

To view, change, or configure systemd targets, use the `systemctl` utility as described in [Table 6.7, “Comparison of SysV init Commands with systemctl”](#) and in the sections below. The `runlevel` and `telinit` commands are still available in the system and work as expected, but are only included for compatibility reasons and should be avoided.

Table 6.7. Comparison of SysV init Commands with systemctl

Old Command	New Command	Description
<code>runlevel</code>	<code>systemctl list-units --type target</code>	Lists currently loaded target units.
<code>telinit runlevel</code>	<code>systemctl isolate name.target</code>	Changes the current target.

6.3.1. Viewing the Default Target

To determine which target unit is used by default, run the following command:

```
systemctl get-default
```

This command resolves the symbolic link located at `/etc/systemd/system/default.target` and displays the result. For information on how to change the default target, see [Section 6.3.3, “Changing the Default Target”](#). For information on how to list all currently loaded target units, see [Section 6.3.2, “Viewing the Current Target”](#).

Example 6.8. Viewing the Default Target

To display the default target unit, type:

```
~]$ systemctl get-default  
graphical.target
```

6.3.2. Viewing the Current Target

To list all currently loaded target units, type the following command at a shell prompt:

```
systemctl list-units --type target
```

For each target unit, this command displays its full name (**UNIT**) followed by a note whether the unit has been loaded (**LOAD**), its high-level (**ACTIVE**) and low-level (**SUB**) unit activation state, and a short description (**DESCRIPTION**).

By default, the **systemctl list-units** command displays only active units. If you want to list all loaded units regardless of their state, run this command with the **--all** or **-a** command line option:

```
systemctl list-units --type target --all
```

See [Section 6.3.1, “Viewing the Default Target”](#) for information on how to display the default target. For information on how to change the current target, see [Section 6.3.4, “Changing the Current Target”](#).

Example 6.9. Viewing the Current Target

To list all currently loaded target units, run the following command:

```
~]$ systemctl list-units --type target
UNIT                  LOAD ACTIVE SUB   DESCRIPTION
basic.target           loaded active active Basic System
cryptsetup.target      loaded active active Encrypted Volumes
getty.target            loaded active active Login Prompts
graphical.target       loaded active active Graphical Interface
local-fs-pre.target   loaded active active Local File Systems (Pre)
local-fs.target         loaded active active Local File Systems
multi-user.target      loaded active active Multi-User System
network.target          loaded active active Network
paths.target             loaded active active Paths
remote-fs.target        loaded active active Remote File Systems
sockets.target          loaded active active Sockets
sound.target              loaded active active Sound Card
spice-vdagentd.target  loaded active active Agent daemon for Spice guests
swap.target                loaded active active Swap
sysinit.target           loaded active active System Initialization
time-sync.target         loaded active active System Time Synchronized
timers.target              loaded active active Timers

LOAD    = Reflects whether the unit definition was properly loaded.
ACTIVE  = The high-level unit activation state, i.e. generalization of SUB.
SUB     = The low-level unit activation state, values depend on unit type.

17 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
```

6.3.3. Changing the Default Target

To configure the system to use a different target unit by default, type the following at a shell prompt as **root**:

```
systemctl set-default name.target
```

Replace *name* with the name of the target unit you want to use by default (for example, **multi-user**). This command replaces the **/etc/systemd/system/default.target** file with a symbolic link to **/usr/lib/systemd/system/name.target**, where *name* is the name of the target unit you want to use. For information on how to change the current target, see [Section 6.3.4, “Changing the Current Target”](#).

Example 6.10. Changing the Default Target

To configure the system to use the **multi-user.target** unit by default, run the following command as **root**:

```
~]# systemctl set-default multi-user.target
rm '/etc/systemd/system/default.target'
ln -s '/usr/lib/systemd/system/multi-user.target'
'/etc/systemd/system/default.target'
```

6.3.4. Changing the Current Target

To change to a different target unit in the current session, type the following at a shell prompt as **root**:

```
systemctl isolate name.target
```

Replace *name* with the name of the target unit you want to use (for example, **multi-user**). This command starts the target unit named *name* and all dependent units, and immediately stops all others. For information on how to change the default target, see [Section 6.3.3, “Changing the Default Target”](#). For information on how to list all currently loaded target units, see [Section 6.3.4, “Changing the Current Target”](#).

Example 6.11. Changing the Current Target

To turn off the graphical user interface and change to the **multi-user.target** unit in the current session, run the following command as **root**:

```
~]# systemctl isolate multi-user.target
```

6.3.5. Changing to Rescue Mode

Rescue mode provides a convenient single-user environment and allows you to repair your system in situations when it is unable to complete a regular booting process. In rescue mode, the system attempts to mount all local file systems and start some important system services, but it does not activate network interfaces or allow more users to be logged into the system at the same time. In Red Hat Enterprise Linux 7, rescue mode is equivalent to *single user mode* and requires the root password.

To change the current target and enter rescue mode in the current session, type the following at a shell prompt as **root**:

```
systemctl rescue
```

This command is similar to **systemctl isolate rescue.target**, but it also sends an informative message to all users that are currently logged into the system. To prevent systemd from sending this message, run this command with the **--no-wall** command line option:

```
systemctl --no-wall rescue
```

For information on how to enter emergency mode, see [Section 6.3.6, “Changing to Emergency Mode”](#).

Example 6.12. Changing to Rescue Mode

To enter rescue mode in the current session, run the following command as **root**:

```
~]# systemctl rescue
```

```
Broadcast message from root@localhost on pts/0 (Fri 2013-10-25 18:23:15 CEST):
```

```
The system is going down to rescue mode NOW!
```

6.3.6. Changing to Emergency Mode

Emergency mode provides the most minimal environment possible and allows you to repair your system even in situations when the system is unable to enter rescue mode. In emergency mode, the system

mounts the root file system only for reading, does not attempt to mount any other local file systems, does not activate network interfaces, and only starts few essential services. In Red Hat Enterprise Linux 7, emergency mode requires the root password.

To change the current target and enter emergency mode, type the following at a shell prompt as **root**:

```
systemctl emergency
```

This command is similar to **systemctl isolate emergency.target**, but it also sends an informative message to all users that are currently logged into the system. To prevent systemd from sending this message, run this command with the **--no-wall** command line option:

```
systemctl --no-wall emergency
```

For information on how to enter rescue mode, see [Section 6.3.5, “Changing to Rescue Mode”](#).

Example 6.13. Changing to Emergency Mode

To enter emergency mode without sending a message to all users that are currently logged into the system, run the following command as **root**:

```
~]# systemctl --no-wall emergency
```

6.4. Shutting Down, Suspending, and Hibernating the System

In Red Hat Enterprise Linux 7, the **systemctl** utility replaces a number of power management commands used in previous versions of the Red Hat Enterprise Linux system. The commands listed in [Table 6.8, “Comparison of Power Management Commands with systemctl”](#) are still available in the system for compatibility reasons, but it is advised that you use **systemctl** when possible.

Table 6.8. Comparison of Power Management Commands with systemctl

Old Command	New Command	Description
halt	systemctl halt	Halts the system.
poweroff	systemctl poweroff	Powers off the system.
reboot	systemctl reboot	Restarts the system.
pm-suspend	systemctl suspend	Suspends the system.
pm-hibernate	systemctl hibernate	Hibernates the system.
pm-suspend-hybrid	systemctl hybrid-sleep	Hibernates and suspends the system.

6.4.1. Shutting Down the System

To shut down the system and power off the machine, type the following at a shell prompt as **root**:

```
systemctl poweroff
```

To shut down and halt the system without powering off the machine, run the following command as **root**:

```
systemctl halt
```

By default, running either of these commands causes systemd to send an informative message to all users that are currently logged into the system. To prevent systemd from sending this message, run the selected command with the **--no-wall** command line option, for example:

```
systemctl --no-wall poweroff
```

6.4.2. Restarting the System

To restart the system, run the following command as **root**:

```
systemctl reboot
```

By default, this command causes systemd to send an informative message to all users that are currently logged into the system. To prevent systemd from sending this message, run this command with the **--no-wall** command line option:

```
systemctl --no-wall reboot
```

6.4.3. Suspending the System

To suspend the system, type the following at a shell prompt as **root**:

```
systemctl suspend
```

This command saves the system state in RAM and with the exception of the RAM module, powers off most of the devices in the machine. When you turn the machine back on, the system then restores its state from RAM without having to boot again. Because the system state is saved in RAM and not on the hard disk, restoring the system from suspend mode is significantly faster than restoring it from hibernation, but as a consequence, a suspended system state is also vulnerable to power outages.

For information on how to hibernate the system, see [Section 6.4.4, “Hibernating the System”](#).

6.4.4. Hibernating the System

To hibernate the system, type the following at a shell prompt as **root**:

```
systemctl hibernate
```

This command saves the system state on the hard disk drive and powers off the machine. When you turn the machine back on, the system then restores its state from the saved data without having to boot again. Because the system state is saved on the hard disk and not in RAM, the machine does not have to maintain electrical power to the RAM module, but as a consequence, restoring the system from hibernation is significantly slower than restoring it from suspend mode.

To hibernate and suspend the system, run the following command as **root**:

```
systemctl hybrid-sleep
```

For information on how to suspend the system, see [Section 6.4.3, “Suspending the System”](#).

6.5. Controlling systemd on a Remote Machine

In addition to controlling the systemd system and service manager locally, the **systemctl** utility also allows you to interact with systemd running on a remote machine over the SSH protocol. Provided that the **sshd** service on the remote machine is running, you can connect to this machine by running the **systemctl** command with the **--host** or **-H** command line option:

```
systemctl --host user_name@host_name command
```

Replace *user_name* with the name of the remote user, *host_name* with the machine's host name, and **command** with any of the **systemctl** commands described above. Note that the remote machine must be configured to allow the selected user remote access over the SSH protocol. For more information on how to configure an SSH server, see [Chapter 7, OpenSSH](#).

Example 6.14. Remote Management

To log in to a remote machine named **server-01.example.com** as the **root** user and determine the current status of the **httpd.service** unit, type the following at a shell prompt:

```
~]$ systemctl -H root@server-01.example.com status httpd.service
root@server-01.example.com's password:
httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled)
   Active: active (running) since Fri 2013-11-01 13:58:56 CET; 2h 48min ago
     Main PID: 649
        Status: "Total requests: 0; Current requests/sec: 0; Current traffic: 0 B/sec"
       CGroup: /system.slice/httpd.service
```

6.6. Additional Resources

For more information on systemd and its usage on Red Hat Enterprise Linux, see the resources listed below.

Installed Documentation

- ▶ **systemctl(1)** — The manual page for the **systemctl** command line utility provides a complete list of supported options and commands.
- ▶ **systemd(1)** — The manual page for the **systemd** system and service manager provides more information about its concepts and documents available command line options and environment variables, supported configuration files and directories, recognized signals, and available kernel options.
- ▶ **systemd.unit(5)** — The manual page named **systemd.unit** provides in-depth information about systemd unit files and documents all available configuration options.
- ▶ **systemd.service(5)** — The manual page named **systemd.service** documents the format of service unit files.
- ▶ **systemd.target(5)** — The manual page named **systemd.target** documents the format of target unit files.

Online Documentation

- ▶ [Red Hat Enterprise Linux 7 Networking Guide](#) — The *Networking Guide* for Red Hat Enterprise Linux 7 documents relevant information regarding the configuration and administration of network interfaces, networks, and network services in this system. It provides an introduction to the `hostnamectl` utility, explains how to use it to view and set host names on the command line, both locally and remotely, and provides important information about the selection of host names and domain names.
- ▶ [Red Hat Enterprise Linux 7 Desktop Migration and Administration Guide](#) — The *Desktop Migration and Administration Guide* for Red Hat Enterprise Linux 7 documents the migration planning, deployment, configuration, and administration of the GNOME 3 desktop on this system. It introduces the `logind` service, enumerates its most significant features, and explains how to use the `loginctl` utility to list active sessions and enable multi-seat support.
- ▶ [Red Hat Enterprise Linux 7 SELinux User's and Administrator's Guide](#) — The *SELinux User's and Administrator's Guide* for Red Hat Enterprise Linux 7 describes the basic principles of SELinux and documents in detail how to configure and use SELinux with various services such as the Apache HTTP Server, Postfix, PostgreSQL, or OpenShift. It explains how to configure SELinux access permissions for system services managed by `systemd`.
- ▶ [Red Hat Enterprise Linux 7 Installation Guide](#) — The *Installation Guide* for Red Hat Enterprise Linux 7 documents how to install the system on AMD64 and Intel 64 systems, 64-bit IBM Power Systems servers, and IBM System z. It also covers advanced installation methods such as Kickstart installations, PXE installations, and installations over the VNC protocol. In addition, it describes common post-installation tasks and explains how to troubleshoot installation problems, including detailed instructions on how to boot into rescue mode or recover the root password.
- ▶ [Red Hat Enterprise Linux 7 Security Guide](#) — The *Security Guide* for Red Hat Enterprise Linux 7 assists users and administrators in learning the processes and practices of securing their workstations and servers against local and remote intrusion, exploitation, and malicious activity. It also explains how to secure critical system services.
- ▶ [systemd Home Page](#) — The project home page provides more information about `systemd`.

See Also

- ▶ [Chapter 1, System Locale and Keyboard Configuration](#) documents how to manage the system locale and keyboard layouts. It explains how to use the `localectl` utility to view the current locale, list available locales, and set the system locale on the command line, as well as to view the current keyboard layout, list available keymaps, and enable a particular keyboard layout on the command line.
- ▶ [Chapter 2, Configuring the Date and Time](#) documents how to manage the system date and time. It explains the difference between a real-time clock and system clock and describes how to use the `timedatectl` utility to display the current settings of the system clock, configure the date and time, change the time zone, and synchronize the system clock with a remote server.
- ▶ [Chapter 4, Gaining Privileges](#) documents how to gain administrative privileges by using the `su` and `sudo` commands.
- ▶ [Chapter 7, OpenSSH](#) describes how to configure an SSH server and how to use the `ssh`, `scp`, and `sftp` client utilities to access it.
- ▶ [Chapter 18, Viewing and Managing Log Files](#) provides an introduction to `journald`. It describes the journal, introduces the `journald` service, and documents how to use the `journalctl` utility to view log entries, enter live view mode, and filter log entries. In addition, this chapter describes how to give non-root users access to system logs and enable persistent storage for log files.

Chapter 7. OpenSSH

SSH (Secure Shell) is a protocol which facilitates secure communications between two systems using a client/server architecture and allows users to log in to server host systems remotely. Unlike other remote communication protocols, such as **FTP** or **Telnet**, SSH encrypts the login session, rendering the connection difficult for intruders to collect unencrypted passwords.

The **ssh** program is designed to replace older, less secure terminal applications used to log in to remote hosts, such as **telnet** or **rsh**. A related program called **scp** replaces older programs designed to copy files between hosts, such as **rcp**. Because these older applications do not encrypt passwords transmitted between the client and the server, avoid them whenever possible. Using secure methods to log in to remote systems decreases the risks for both the client system and the remote host.

Red Hat Enterprise Linux includes the general OpenSSH package, *openssh*, as well as the OpenSSH server, *openssh-server*, and client, *openssh-clients*, packages. Note, the OpenSSH packages require the OpenSSL package *openssl-libs*, which installs several important cryptographic libraries, enabling OpenSSH to provide encrypted communications.

7.1. The SSH Protocol

7.1.1. Why Use SSH?

Potential intruders have a variety of tools at their disposal enabling them to disrupt, intercept, and re-route network traffic in an effort to gain access to a system. In general terms, these threats can be categorized as follows:

Interception of communication between two systems

The attacker can be somewhere on the network between the communicating parties, copying any information passed between them. He may intercept and keep the information, or alter the information and send it on to the intended recipient.

This attack is usually performed using a *packet sniffer*, a rather common network utility that captures each packet flowing through the network, and analyzes its content.

Impersonation of a particular host

Attacker's system is configured to pose as the intended recipient of a transmission. If this strategy works, the user's system remains unaware that it is communicating with the wrong host.

This attack can be performed using a technique known as *DNS poisoning*, or via so-called *IP spoofing*. In the first case, the intruder uses a cracked DNS server to point client systems to a maliciously duplicated host. In the second case, the intruder sends falsified network packets that appear to be from a trusted host.

Both techniques intercept potentially sensitive information and, if the interception is made for hostile reasons, the results can be disastrous. If SSH is used for remote shell login and file copying, these security threats can be greatly diminished. This is because the SSH client and server use digital signatures to verify their identity. Additionally, all communication between the client and server systems is encrypted. Attempts to spoof the identity of either side of a communication does not work, since each packet is encrypted using a key known only by the local and remote systems.

7.1.2. Main Features

The SSH protocol provides the following safeguards:

No one can pose as the intended server

After an initial connection, the client can verify that it is connecting to the same server it had connected to previously.

No one can capture the authentication information

The client transmits its authentication information to the server using strong, 128-bit encryption.

No one can intercept the communication

All data sent and received during a session is transferred using 128-bit encryption, making intercepted transmissions extremely difficult to decrypt and read.

Additionally, it also offers the following options:

It provides secure means to use graphical applications over a network

Using a technique called *X11 forwarding*, the client can forward *X Window System* applications from the server.

It provides a way to secure otherwise insecure protocols

The SSH protocol encrypts everything it sends and receives. Using a technique called *port forwarding*, an SSH server can become a conduit to securing otherwise insecure protocols, like POP, and increasing overall system and data security.

It can be used to create a secure channel

The OpenSSH server and client can be configured to create a tunnel similar to a virtual private network for traffic between server and client machines.

It supports the Kerberos authentication

OpenSSH servers and clients can be configured to authenticate using the GSSAPI (Generic Security Services Application Program Interface) implementation of the Kerberos network authentication protocol.

7.1.3. Protocol Versions

Two varieties of SSH currently exist: version 1, and newer version 2. The OpenSSH suite under Red Hat Enterprise Linux uses SSH version 2, which has an enhanced key exchange algorithm not vulnerable to the known exploit in version 1. However, for compatibility reasons, the OpenSSH suite does support version 1 connections as well.



Important

To ensure maximum security for your connection, it is recommended that only SSH version 2-compatible servers and clients are used whenever possible.

7.1.4. Event Sequence of an SSH Connection

The following series of events help protect the integrity of SSH communication between two hosts.

1. A cryptographic handshake is made so that the client can verify that it is communicating with the correct server.
2. The transport layer of the connection between the client and remote host is encrypted using a

symmetric cipher.

3. The client authenticates itself to the server.
4. The remote client interacts with the remote host over the encrypted connection.

7.1.4.1. Transport Layer

The primary role of the transport layer is to facilitate safe and secure communication between the two hosts at the time of authentication and during subsequent communication. The transport layer accomplishes this by handling the encryption and decryption of data, and by providing integrity protection of data packets as they are sent and received. The transport layer also provides compression, speeding the transfer of information.

Once an SSH client contacts a server, key information is exchanged so that the two systems can correctly construct the transport layer. The following steps occur during this exchange:

- ▶ Keys are exchanged
- ▶ The public key encryption algorithm is determined
- ▶ The symmetric encryption algorithm is determined
- ▶ The message authentication algorithm is determined
- ▶ The hash algorithm is determined

During the key exchange, the server identifies itself to the client with a unique *host key*. If the client has never communicated with this particular server before, the server's host key is unknown to the client and it does not connect. OpenSSH gets around this problem by accepting the server's host key. This is done after the user is notified and has both accepted and verified the new host key. In subsequent connections, the server's host key is checked against the saved version on the client, providing confidence that the client is indeed communicating with the intended server. If, in the future, the host key no longer matches, the user must remove the client's saved version before a connection can occur.



Warning

It is possible for an attacker to masquerade as an SSH server during the initial contact since the local system does not know the difference between the intended server and a false one set up by an attacker. To help prevent this, verify the integrity of a new SSH server by contacting the server administrator before connecting for the first time or in the event of a host key mismatch.

SSH is designed to work with almost any kind of public key algorithm or encoding format. After an initial key exchange creates a hash value used for exchanges and a shared secret value, the two systems immediately begin calculating new keys and algorithms to protect authentication and future data sent over the connection.

After a certain amount of data has been transmitted using a given key and algorithm (the exact amount depends on the SSH implementation), another key exchange occurs, generating another set of hash values and a new shared secret value. Even if an attacker is able to determine the hash and shared secret value, this information is only useful for a limited period of time.

7.1.4.2. Authentication

Once the transport layer has constructed a secure tunnel to pass information between the two systems, the server tells the client the different authentication methods supported, such as using a private key-encoded signature or typing a password. The client then tries to authenticate itself to the server using one of these supported methods.

SSH servers and clients can be configured to allow different types of authentication, which gives each side the optimal amount of control. The server can decide which encryption methods it supports based on its security model, and the client can choose the order of authentication methods to attempt from the available options.

7.1.4.3. Channels

After a successful authentication over the SSH transport layer, multiple channels are opened via a technique called *multiplexing* [2]. Each of these channels handles communication for different terminal sessions and for forwarded X11 sessions.

Both clients and servers can create a new channel. Each channel is then assigned a different number on each end of the connection. When the client attempts to open a new channel, the client sends the channel number along with the request. This information is stored by the server and is used to direct communication to that channel. This is done so that different types of sessions do not affect one another and so that when a given session ends, its channel can be closed without disrupting the primary SSH connection.

Channels also support *flow-control*, which allows them to send and receive data in an orderly fashion. In this way, data is not sent over the channel until the client receives a message that the channel is open.

The client and server negotiate the characteristics of each channel automatically, depending on the type of service the client requests and the way the user is connected to the network. This allows great flexibility in handling different types of remote connections without having to change the basic infrastructure of the protocol.

7.2. Configuring OpenSSH

7.2.1. Configuration Files

There are two different sets of configuration files: those for client programs (that is, **ssh**, **scp**, and **sftp**), and those for the server (the **sshd** daemon).

System-wide SSH configuration information is stored in the **/etc/ssh/** directory as described in [Table 7.1, “System-wide configuration files”](#). User-specific SSH configuration information is stored in **~/.ssh/** within the user's home directory as described in [Table 7.2, “User-specific configuration files”](#).

Table 7.1. System-wide configuration files

File	Description
/etc/ssh/moduli	Contains Diffie-Hellman groups used for the Diffie-Hellman key exchange which is critical for constructing a secure transport layer. When keys are exchanged at the beginning of an SSH session, a shared, secret value is created which cannot be determined by either party alone. This value is then used to provide host authentication.
/etc/ssh/ssh_config	The default SSH client configuration file. Note that it is overridden by ~/.ssh/config if it exists.
/etc/ssh/sshd_config	The configuration file for the sshd daemon.

File	Description
<code>/etc/ssh/ssh_host_ecdsa_key</code>	The ECDSA private key used by the sshd daemon.
<code>/etc/ssh/ssh_host_ecdsa_key.pub</code>	The ECDSA public key used by the sshd daemon.
<code>/etc/ssh/ssh_host_key</code>	The RSA private key used by the sshd daemon for version 1 of the SSH protocol.
<code>/etc/ssh/ssh_host_key.pub</code>	The RSA public key used by the sshd daemon for version 1 of the SSH protocol.
<code>/etc/ssh/ssh_host_rsa_key</code>	The RSA private key used by the sshd daemon for version 2 of the SSH protocol.
<code>/etc/ssh/ssh_host_rsa_key.pub</code>	The RSA public key used by the sshd daemon for version 2 of the SSH protocol.
<code>/etc/pam.d/sshd</code>	The PAM configuration file for the sshd daemon.
<code>/etc/sysconfig/sshd</code>	Configuration file for the sshd service.

Table 7.2. User-specific configuration files

File	Description
<code>~/.ssh/authorized_keys</code>	Holds a list of authorized public keys for servers. When the client connects to a server, the server authenticates the client by checking its signed public key stored within this file.
<code>~/.ssh/id_ecdsa</code>	Contains the ECDSA private key of the user.
<code>~/.ssh/id_ecdsa.pub</code>	The ECDSA public key of the user.
<code>~/.ssh/id_rsa</code>	The RSA private key used by ssh for version 2 of the SSH protocol.
<code>~/.ssh/id_rsa.pub</code>	The RSA public key used by ssh for version 2 of the SSH protocol.
<code>~/.ssh/identity</code>	The RSA private key used by ssh for version 1 of the SSH protocol.
<code>~/.ssh/identity.pub</code>	The RSA public key used by ssh for version 1 of the SSH protocol.
<code>~/.ssh/known_hosts</code>	Contains host keys of SSH servers accessed by the user. This file is very important for ensuring that the SSH client is connecting to the correct SSH server.

For information concerning various directives that can be used in the SSH configuration files, see the `ssh_config(5)` and `sshd_config(5)` manual pages.

7.2.2. Starting an OpenSSH Server

In order to run an OpenSSH server, you must have the `openssh-server` package installed (see [Section 5.2.4, “Installing Packages”](#) for more information on how to install new packages in Red Hat Enterprise Linux 7).

To start the **sshd** daemon in the current session, type the following at a shell prompt as **root**:

```
~]# systemctl start sshd.service
```

To stop the running **sshd** daemon in the current session, use the following command as **root**:

```
~]# systemctl stop sshd.service
```

If you want the daemon to start automatically at the boot time, type as **root**:

```
~]# systemctl enable sshd.service
ln -s '/usr/lib/systemd/system/sshd.service' '/etc/systemd/system/multi-
user.target.wants/sshd.service'
```

For more information on how to manage system services in Red Hat Enterprise Linux, see [Chapter 6, "Managing Services with systemd"](#).

Note that if you reinstall the system, a new set of identification keys will be created. As a result, clients who had connected to the system with any of the OpenSSH tools before the reinstall will see the following message:

```
@@@@@@@00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!    @
@@@@@@@000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
```

To prevent this, you can backup the relevant files from the **/etc/ssh/** directory (see [Table 7.1, "System-wide configuration files"](#) for a complete list), and restore them whenever you reinstall the system.

7.2.3. Requiring SSH for Remote Connections

For SSH to be truly effective, using insecure connection protocols should be prohibited. Otherwise, a user's password may be protected using SSH for one session, only to be captured later while logging in using Telnet. Some services to disable include **telnet**, **rsh**, **rlogin**, and **vsftpd**.

For information on how to configure the **vsftpd** service, see [Section 12.2, "FTP"](#). To learn how to manage system services in Red Hat Enterprise Linux 7, read [Chapter 6, "Managing Services with systemd"](#).

7.2.4. Using Key-based Authentication

To improve the system security even further, generate SSH key pairs and then enforce key-based authentication by disabling password authentication. To do so, open the **/etc/ssh/sshd_config** configuration file in a text editor such as **vi** or **nano**, and change the **PasswordAuthentication** option as follows:

```
PasswordAuthentication no
```

If you are working on a system other than a new default installation, check that **PubkeyAuthentication no** has **not** been set. If connected remotely, not using console or out-of-band access, testing the key-based log in process before disabling password authentication is advised.

To be able to use **ssh**, **scp**, or **sftp** to connect to the server from a client machine, generate an authorization key pair by following the steps below. Note that keys must be generated for each user separately.

Red Hat Enterprise Linux 7 uses SSH Protocol 2 and RSA keys by default (see [Section 7.1.3, "Protocol Versions"](#) for more information).

**Important**

If you complete the steps as **root**, only **root** will be able to use the keys.

**Note**

If you reinstall your system and want to keep previously generated key pairs, backup the `~/.ssh/` directory. After reinstalling, copy it back to your home directory. This process can be done for all users on your system, including **root**.

7.2.4.1. Generating Key Pairs

To generate an RSA key pair for version 2 of the SSH protocol, follow these steps:

1. Generate an RSA key pair by typing the following at a shell prompt:

```
~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/john/.ssh/id_rsa):
```

2. Press **Enter** to confirm the default location (that is, `~/.ssh/id_rsa`) for the newly created key.
3. Enter a passphrase, and confirm it by entering it again when prompted to do so. For security reasons, avoid using the same password as you use to log in to your account.

After this, you will be presented with a message similar to this:

```
Your identification has been saved in /home/john/.ssh/id_rsa.
Your public key has been saved in /home/john/.ssh/id_rsa.pub.
The key fingerprint is:
e7:97:c7:e2:0e:f9:0e:fc:c4:d7:cb:e5:31:11:92:14 john@penguin.example.com
The key's randomart image is:
+--[ RSA 2048]----+
|   E. |
|   . . |
|   o . |
|   . . |
|   S . . |
| + o o ..|
| * * +oo |
| 0 +..=|
| 0* o. |
+-----+
```

4. Change the permissions of the `~/.ssh/` directory:

```
~]$ chmod 700 ~/.ssh
```

5. To copy the public key to a remote machine, issue a command in the following format:

```
ssh-copy-id user@hostname
```

This will copy all `~/.ssh/id* .pub` public keys. Alternatively, specify the public keys file name as follows:

```
ssh-copy-id -i ~/.ssh/id_rsa.pub user@hostname
```

This will copy the content of `~/.ssh/id_rsa.pub` into the `~/.ssh/authorized_keys` file on the machine to which you want to connect. If the file already exists, the keys are appended to its end.

To generate an ECDSA key pair for version 2 of the SSH protocol, follow these steps:

1. Generate an ECDSA key pair by typing the following at a shell prompt:

```
~]$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/john/.ssh/id_ecdsa):
```

2. Press **Enter** to confirm the default location (that is, `~/.ssh/id_ecdsa`) for the newly created key.
3. Enter a passphrase, and confirm it by entering it again when prompted to do so. For security reasons, avoid using the same password as you use to log in to your account.

After this, you will be presented with a message similar to this:

```
Your identification has been saved in /home/john/.ssh/id_ecdsa.
Your public key has been saved in /home/john/.ssh/id_ecdsa.pub.
The key fingerprint is:
fd:1d:ca:10:52:96:21:43:7e:bd:4c:fc:5b:35:6b:63 sjw@my-server
The key's randomart image is:
+--[ECDSA 256]---+
|       .+ +o      |
|       . =.o       |
|       o o + ..   |
|       + + o +    |
|       S o o oE.   |
|           + oo+.  |
|           + o     |
|                   |
+-----+
```

4. Change the permissions of the `~/.ssh/` directory:

```
~]$ chmod 700 ~/.ssh
```

5. To copy the public key to a remote machine, issue a command in the following format:

```
ssh-copy-id user@hostname
```

This will copy all `~/.ssh/id* .pub` public keys. Alternatively, specify the public keys file name as follows:

```
ssh-copy-id -i ~/.ssh/id_ecdsa.pub user@hostname
```

This will copy the content of `~/.ssh/id_ecdsa.pub` into the `~/.ssh/authorized_keys` on the machine to which you want to connect. If the file already exists, the keys are appended to its end.

See [Section 7.2.4.2, “Configuring ssh-agent”](#) for information on how to set up your system to remember the passphrase.



Important

The private key is for your personal use only, and it is important that you never give it to anyone.

7.2.4.2. Configuring ssh-agent

To store your passphrase so that you do not have to enter it each time you initiate a connection with a remote machine, you can use the **ssh-agent** authentication agent. If you are running GNOME, you can configure it to prompt you for your passphrase whenever you log in and remember it during the whole session. Otherwise you can store the passphrase for a certain shell prompt.

To save your passphrase during your GNOME session, follow these steps:

1. Make sure you have the *openssh-askpass* package installed. If not, see [Section 5.2.4, “Installing Packages”](#) for more information on how to install new packages in Red Hat Enterprise Linux.
2. Press the **Super** key to enter the Activities Overview, type **Startup Applications** and then press **Enter**. The **Startup Applications Preferences** tool appears. The tab containing a list of available startup programs will be shown by default.

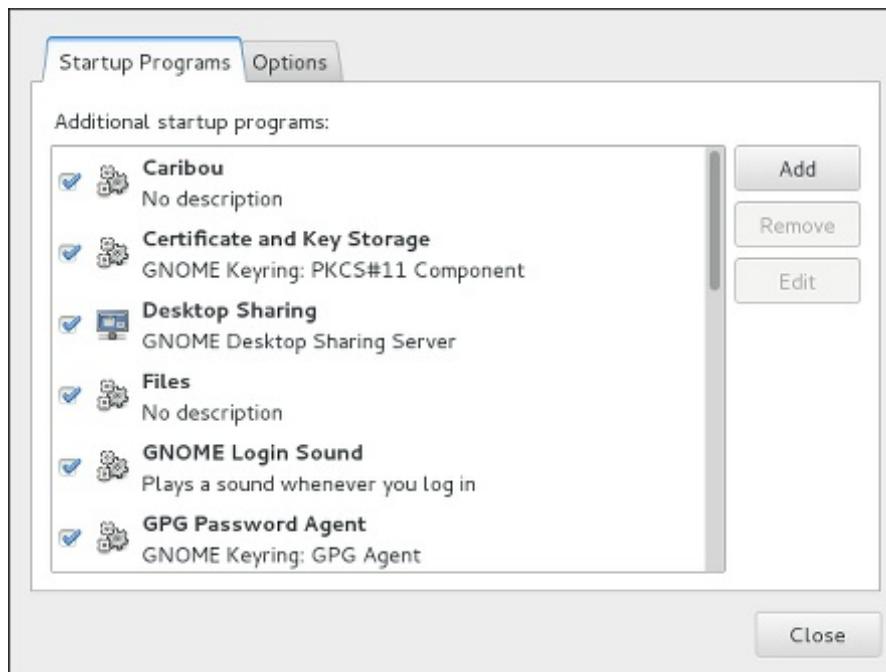


Figure 7.1. Startup Applications Preferences

3. Click the **Add** button on the right, and enter **/usr/bin/ssh-add** in the **Command** field.

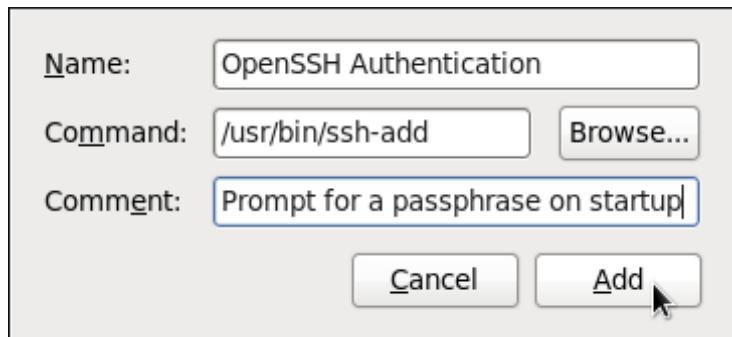


Figure 7.2. Adding new application

- Click **Add** and make sure the checkbox next to the newly added item is selected.

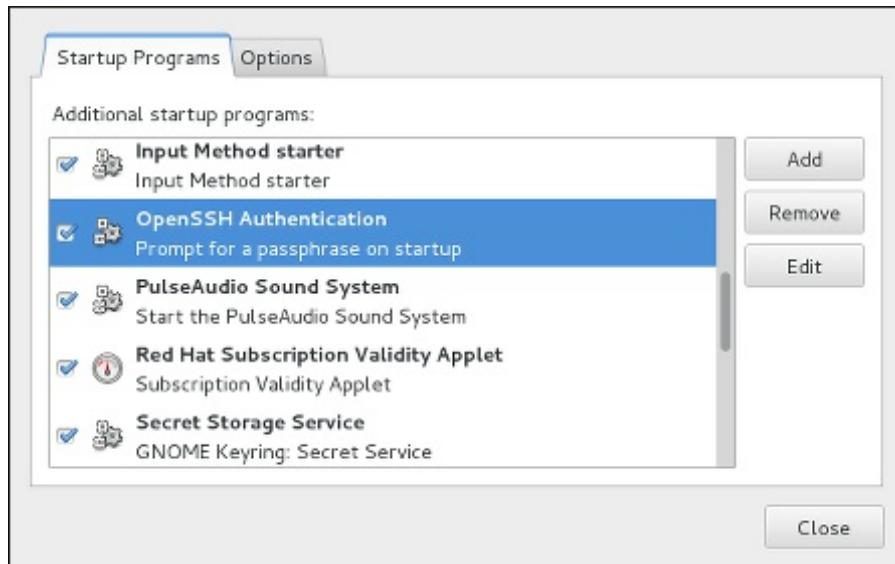


Figure 7.3. Enabling the application

- Log out and then log back in. A dialog box will appear prompting you for your passphrase. From this point on, you should not be prompted for a password by **ssh**, **scp**, or **sftp**.

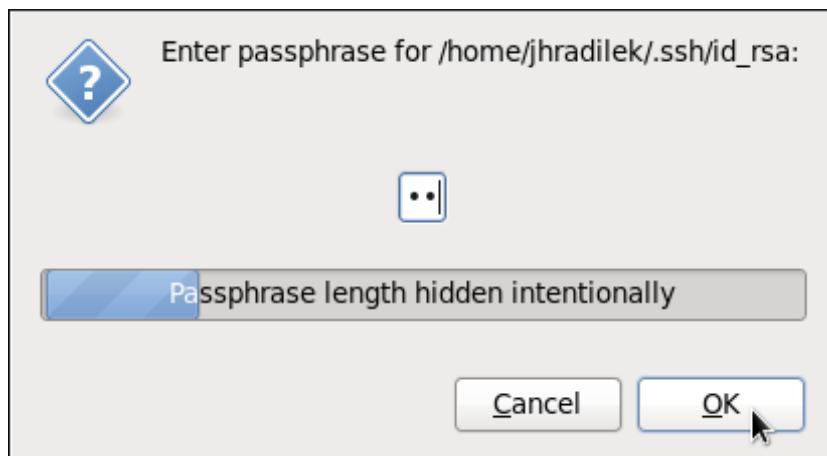


Figure 7.4. Entering a passphrase

To save your passphrase for a certain shell prompt, use the following command:

```
~]$ ssh-add
Enter passphrase for /home/john/.ssh/id_rsa:
```

Note that when you log out, your passphrase will be forgotten. You must execute the command each time you log in to a virtual console or a terminal window.

7.3. OpenSSH Clients

To connect to an OpenSSH server from a client machine, you must have the *openssh-clients* package installed (see to [Section 5.2.4, “Installing Packages”](#) for more information on how to install new packages in Red Hat Enterprise Linux).

7.3.1. Using the ssh Utility

The **ssh** utility allows you to log in to a remote machine and execute commands there. It is a secure replacement for the **rlogin**, **rsh**, and **telnet** programs.

Similarly to the **telnet** command, log in to a remote machine by using the following command:

```
ssh hostname
```

For example, to log in to a remote machine named **penguin.example.com**, type the following at a shell prompt:

```
~]$ ssh penguin.example.com
```

This will log you in with the same user name you are using on the local machine. If you want to specify a different user name, use a command in the following form:

```
ssh username@hostname
```

For example, to log in to **penguin.example.com** as **john**, type:

```
~]$ ssh john@penguin.example.com
```

The first time you initiate a connection, you will be presented with a message similar to this:

```
The authenticity of host 'penguin.example.com' can't be established.
ECDSA key fingerprint is 256 da:24:43:0b:2e:c1:3f:a1:84:13:92:01:52:b4:84:ff.
Are you sure you want to continue connecting (yes/no)?
```

Users should always check if the fingerprint is correct before answering the question in this dialog. The user can ask the administrator of the server to confirm the key is correct. This should be done in a secure and previously agreed way. If the user has access to the server's host keys, the fingerprint can be checked by using the **ssh-keygen** command as follows:

```
~]# ssh-keygen -l -f /etc/ssh/ssh_host_ecdsa_key.pub
256 da:24:43:0b:2e:c1:3f:a1:84:13:92:01:52:b4:84:ff      (ECDSA)
```

Type **yes** to accept the key and confirm the connection. You will see a notice that the server has been added to the list of known hosts, and a prompt asking for your password:

Warning: Permanently added 'penguin.example.com' (ECDSA) to the list of known hosts.
john@penguin.example.com's password:



Important

If the SSH server's host key changes, the client notifies the user that the connection cannot proceed until the server's host key is deleted from the `~/.ssh/known_hosts` file. Before doing this, however, contact the system administrator of the SSH server to verify the server is not compromised.

To remove a key from the `~/.ssh/known_hosts` file, issue a command as follows:

```
~]# ssh-keygen-Rpenguin.example.com
# Host penguin.example.com found: line 15 type ECDSA
/home/john/.ssh/known_hosts updated.
Original contents retained as /home/john/.ssh/known_hosts.old
```

After entering the password, you will be provided with a shell prompt for the remote machine.

Alternatively, the `ssh` program can be used to execute a command on the remote machine without logging in to a shell prompt:

```
ssh [username@]hostname command
```

For example, the `/etc/redhat-release` file provides information about the Red Hat Enterprise Linux version. To view the contents of this file on `penguin.example.com`, type:

```
~]$ ssh john@penguin.example.com cat /etc/redhat-release
john@penguin.example.com's password:
Red Hat Enterprise Linux Server release 7.0 (Maipo)
```

After you enter the correct password, the user name will be displayed, and you will return to your local shell prompt.

7.3.2. Using the `scp` Utility

`scp` can be used to transfer files between machines over a secure, encrypted connection. In its design, it is very similar to `rcp`.

To transfer a local file to a remote system, use a command in the following form:

```
scp localfile username@hostname:remotefile
```

For example, if you want to transfer `taglist.vim` to a remote machine named `penguin.example.com`, type the following at a shell prompt:

```
~]$ scp taglist.vim john@penguin.example.com:.vim/plugin/taglist.vim
john@penguin.example.com's password:
taglist.vim                                         100%   144KB 144.5KB/s   00:00
```

Multiple files can be specified at once. To transfer the contents of `.vim/plugin/` to the same directory on the remote machine `penguin.example.com`, type the following command:

```
~]$ scp .vim/plugin/* john@penguin.example.com:.vim/plugin/
john@penguin.example.com's password:
closetag.vim                      100%   13KB  12.6KB/s  00:00
snippetsEmu.vim                    100%   33KB  33.1KB/s  00:00
taglist.vim                       100%  144KB 144.5KB/s  00:00
```

To transfer a remote file to the local system, use the following syntax:

```
scp username@hostname:remotefile localfile
```

For instance, to download the `.vimrc` configuration file from the remote machine, type:

```
~]$ scp john@penguin.example.com:.vimrc .vimrc
john@penguin.example.com's password:
.vimrc                                100% 2233      2.2KB/s  00:00
```

7.3.3. Using the sftp Utility

The `sftp` utility can be used to open a secure, interactive FTP session. In its design, it is similar to `ftp` except that it uses a secure, encrypted connection.

To connect to a remote system, use a command in the following form:

```
sftp username@hostname
```

For example, to log in to a remote machine named `penguin.example.com` with `john` as a username, type:

```
~]$ sftp john@penguin.example.com
john@penguin.example.com's password:
Connected to penguin.example.com.
sftp>
```

After you enter the correct password, you will be presented with a prompt. The `sftp` utility accepts a set of commands similar to those used by `ftp` (see [Table 7.3, “A selection of available sftp commands”](#)).

Table 7.3. A selection of available sftp commands

Command	Description
<code>ls [directory]</code>	List the content of a remote <i>directory</i> . If none is supplied, a current working directory is used by default.
<code>cd directory</code>	Change the remote working directory to <i>directory</i> .
<code>mkdir directory</code>	Create a remote <i>directory</i> .
<code>rmdir path</code>	Remove a remote <i>directory</i> .
<code>put localfile [remotefile]</code>	Transfer <i>localfile</i> to a remote machine.
<code>get remotefile [/localfile]</code>	Transfer <i>remotefile</i> from a remote machine.

For a complete list of available commands, see the `sftp(1)` manual page.

7.4. More Than a Secure Shell

A secure command line interface is just the beginning of the many ways SSH can be used. Given the proper amount of bandwidth, X11 sessions can be directed over an SSH channel. Or, by using TCP/IP forwarding, previously insecure port connections between systems can be mapped to specific SSH channels.

7.4.1. X11 Forwarding

To open an X11 session over an SSH connection, use a command in the following form:

```
ssh -Y username@hostname
```

For example, to log in to a remote machine named **penguin.example.com** with **john** as a username, type:

```
~]$ ssh -Y john@penguin.example.com  
john@penguin.example.com's password:
```

When an X program is run from the secure shell prompt, the SSH client and server create a new secure channel, and the X program data is sent over that channel to the client machine transparently.

X11 forwarding can be very useful. For example, X11 forwarding can be used to create a secure, interactive session of the **Printer Configuration** utility. To do this, connect to the server using **ssh** and type:

```
~]$ system-config-printer &
```

The **Printer Configuration Tool** will appear, allowing the remote user to safely configure printing on the remote system.

7.4.2. Port Forwarding

SSH can secure otherwise insecure **TCP/IP** protocols via port forwarding. When using this technique, the SSH server becomes an encrypted conduit to the SSH client.

Port forwarding works by mapping a local port on the client to a remote port on the server. SSH can map any port from the server to any port on the client. Port numbers do not need to match for this technique to work.



Note

Setting up port forwarding to listen on ports below 1024 requires **root** level access.

To create a TCP/IP port forwarding channel which listens for connections on the **localhost**, use a command in the following form:

```
ssh -L local-port:remote-hostname:remote-port username@hostname
```

For example, to check email on a server called **mail.example.com** using **POP3** through an encrypted connection, use the following command:

```
~]$ ssh -L 1100:mail.example.com:110 mail.example.com
```

Once the port forwarding channel is in place between the client machine and the mail server, direct a POP3 mail client to use port **1100** on the **localhost** to check for new email. Any requests sent to port **1100** on the client system will be directed securely to the **mail.example.com** server.

If **mail.example.com** is not running an SSH server, but another machine on the same network is, SSH can still be used to secure part of the connection. However, a slightly different command is necessary:

```
~]$ ssh -L 1100:mail.example.com:110 other.example.com
```

In this example, POP3 requests from port **1100** on the client machine are forwarded through the SSH connection on port **22** to the SSH server, **other.example.com**. Then, **other.example.com** connects to port **110** on **mail.example.com** to check for new email. Note that when using this technique, only the connection between the client system and **other.example.com** SSH server is secure.

Port forwarding can also be used to get information securely through network firewalls. If the firewall is configured to allow SSH traffic via its standard port (that is, port 22) but blocks access to other ports, a connection between two hosts using the blocked ports is still possible by redirecting their communication over an established SSH connection.



Important

Using port forwarding to forward connections in this manner allows any user on the client system to connect to that service. If the client system becomes compromised, the attacker also has access to forwarded services.

System administrators concerned about port forwarding can disable this functionality on the server by specifying a **No** parameter for the **AllowTcpForwarding** line in **/etc/ssh/sshd_config** and restarting the **sshd** service.

7.5. Additional Resources

For more information on how to configure or connect to an OpenSSH server on Red Hat Enterprise Linux, see the resources listed below.

Installed Documentation

- ▶ **sshd(8)** — The manual page for the **sshd** daemon documents available command line options and provides a complete list of supported configuration files and directories.
- ▶ **ssh(1)** — The manual page for the **ssh** client application provides a complete list of available command line options and supported configuration files and directories.
- ▶ **scp(1)** — The manual page for the **scp** utility provides a more detailed description of this utility and its usage.
- ▶ **sftp(1)** — The manual page for the **sftp** utility.
- ▶ **ssh-keygen(1)** — The manual page for the **ssh-keygen** utility documents in detail how to use it to generate, manage, and convert authentication keys used by **ssh**.
- ▶ **ssh_config(5)** — The manual page named **ssh_config** documents available SSH client configuration options.

- ▶ **sshd_config(5)** — The manual page named **sshd_config** provides a full description of available SSH daemon configuration options.

Online Documentation

- ▶ [OpenSSH Home Page](#) — The OpenSSH home page containing further documentation, frequently asked questions, links to the mailing lists, bug reports, and other useful resources.
- ▶ [OpenSSL Home Page](#) — The OpenSSL home page containing further documentation, frequently asked questions, links to the mailing lists, and other useful resources.

See Also

- ▶ [Chapter 4, Gaining Privileges](#) documents how to gain administrative privileges by using the **su** and **sudo** commands.
- ▶ [Chapter 6, Managing Services with systemd](#) provides more information on systemd and documents how to use the **systemctl** command to manage system services.

[2] A multiplexed connection consists of several signals being sent over a shared, common medium. With SSH, different channels are sent over a common secure connection.

Chapter 8. TigerVNC

TigerVNC (Tiger Virtual Network Computing) is a system of graphical desktop sharing which allows you to remotely control other computers.

TigerVNC works on the client-server network: a **server** shares its output (**vncserver**) and a **client** (**vncviewer**) connects to the server.



Note

Unlike in previous Red Hat Enterprise Linux distributions, current **TigerVNC** uses the **systemd** system management daemon for its configuration. The **/etc/sysconfig/vncserver** configuration file has been replaced by **/lib/systemd/system/vncserver@.service**

8.1. VNC Server

vncserver is a utility which starts a VNC (Virtual Network Computing) desktop. It runs Xvnc with appropriate options and starts a window manager on the VNC desktop. **vncserver** allows users to run totally parallel sessions on a machine which can be accessed by any number of clients from anywhere.

8.1.1. Installing VNC Server

To install the TigerVNC server, run the following command as **root**:

```
# yum install tigervnc-server
```

If you desire to use your TigerVNC as the client as well, run the command, which installs the server component along with the client:

```
# yum install vnc
```

8.1.2. Configuring VNC Server

Procedure 8.1. Configuring the first VNC connection

1. Create a new configuration file named **/lib/systemd/system/vncserver@:*display_number*.service** for each of the display numbers you want to enable. Follow the example below: display number 3 is set, which is included in the configuration file name. You need not create a completely new file, just copy-paste the content of **/lib/systemd/system/vncserver@.service**:



Example 8.1. Creating a configuration file

```
# cp /lib/systemd/system/vncserver@.service
/lib/systemd/system/vncserver@:3.service
```

2. Edit `/lib/systemd/system/vncserver@:display_number.service`, setting the `User` and `ExecStart` arguments as in the example below. Leave the remaining lines of the file unmodified. The `-geometry` argument specifies the size of the VNC desktop to be created; by default, it is to 1024x768.

Example 8.2. Setting the arguments in the configuration file

```
User=joe
ExecStart=/sbin/runuser -l joe -c "/usr/bin/vncserver %i -geometry
1280x1024"
```

3. Save the changes.
4. Update the `systemctl` to ensure the changes are taken into account immediately.

```
# systemctl daemon-reload
```

5. Set the password for the user or users defined in the configuration file.

```
# vncpasswd user
Password:
Verify:
```

Repeat the procedure to set the password for other user or users:

```
# su - user2
```



Important

The stored password is not encrypted securely; anyone who has access to the password file can find the plain-text password.

8.1.3. Starting VNC Server

To start the service with a concrete display number, execute the following command:

```
# systemctl start vncserver@:display_number.service
```

You can also enable the service to start automatically at system start. Every time you log in, `vncserver` is automatically started. As `root`, run

```
# systemctl enable vncserver@:display_number.service
```

At this point, other users are able to use the `vncviewer` program to connect to your server using the display number and password defined.

8.1.4. Terminating VNC session

Similarly to starting the `vncserver` service, you can disable the start of the service automatically at start of your operation system:

```
# systemctl disable vncserver@:display_number.service
```

Or, when your operation system is on, you can stop the service by running the following command :

```
# systemctl stop vncserver@:display_number.service
```

8.2. VNC Viewer

vncviewer is the program which shows the shared graphical user interfaces and controls the server.

For operating the **vncviewer**, there is a pop-up menu containing entries which perform various actions such as switching in and out of full-screen mode or quitting the viewer. Alternatively, you can operate **vncviewer** through the terminal; there is a list of parameters **vncviewer** can be used with which you obtain by typing **vncviewer -h** on the command line.

8.2.1. Installing VNC viewer

To install the TigerVNC Viewer, as **root**, run the following command:

```
# yum install vnc
```

For operating the **vncviewer**, there is a pop-up menu containing entries which perform various actions such as switching in and out of full-screen mode or quitting the viewer. Alternatively, you can operate **vncviewer** through the terminal; there is a list of parameters **vncviewer** can be used with which you obtain by typing **vncviewer -h** on the command line.

8.2.2. Connecting to VNC Server

Once your VNC server is configured, you can connect to it from any VNC server. If you have not done it yet, install the package containing **vncviewer**:

```
# yum install tigervnc-server
```

In order to connect to a VNC server, run **vncviewer** in the format of # **vncviewer machine_name.local_domain:port_number**.

Example 8.3. One client connecting to vncserver

For example, with the IP address 192.168.0.4, display number 3, and machine name *joe*, the command looks as follows:

```
# vncviewer joe 192.168.0.4:3
```

8.2.2.1. Firewall

When using a non-encrypted connection, **firewalld** might block your connection. To work around this obstruction, open a concrete port as described bellow:

**Note**

The default port of VNC server is 5900. To reach the port through which a remote desktop will be accessible, you have to create a sum of this default port and user's assigned display number. For example, for the second port: $2 + 5900 = 5902$

1. Run the following command to see the information concerning **firewalld** settings:

```
$ firewall-cmd --list-all
```

2. Verify whether the chosen port is open by executing:

```
$ firewall-cmd --query-port=port_number/tcp
```

8.2.3. Connecting to VNC Server using SSH

VNC stands for a clear text network protocol, so there is no security against possible attacks on the communication. To make the communication secure, use SSH tunnel to encrypt the communication between VNC server and the client.

1.

```
$ vncviewer -via user@host localhost:display_number
```

Example 8.4. Using the -via option

```
$ vncviewer -via joe@192.168.2.101 127.0.0.1:3
```

2. When you are prompted to, type the password, and confirm by pressing **Enter**.
3. A window with a remote desktop appears on your screen.

For more information concerning OpenSSH, see [Chapter 7, OpenSSH](#).

8.3. Additional Resources

vncserver(1)

The VNC server manual pages.

vncviewer(1)

The VNC viewer manual pages.

passwd(1)

The VNC password manual pages.

Part IV. Servers

This part discusses various topics related to servers such as how to set up a Web server or share files and directories over the network.

Chapter 9. Web Servers

HTTP (Hypertext Transfer Protocol) server, or a *web server*, is a network service that serves content to a client over the web. This typically means web pages, but any other documents can be served as well.

9.1. The Apache HTTP Server

The web server available in Red Hat Enterprise Linux 7 is the Apache HTTP server daemon, **httpd**, an open source web server developed by the [Apache Software Foundation](#). In Red Hat Enterprise Linux 7 the Apache server has been updated to **Apache HTTP Server 2.4**. This section describes the basic configuration of the **httpd** service, and covers some advanced topics such as adding server modules, setting up virtual hosts, or configuring the secure HTTP server.

There are important differences between the Apache HTTP Server 2.4 and version 2.2, and if you are upgrading from a previous release of Red Hat Enterprise Linux, you will need to update the **httpd** service configuration accordingly. This section reviews some of the newly added features, outlines important changes, and guides you through the update of older configuration files.

9.1.1. Notable Changes

The Apache HTTP Server version 2.4 has the following changes:

httpd Service Control

With the migration away from SysV init scripts, server administrators should switch to using the **apachectl** and **systemctl** commands to control the service, in place of the **service** command. The following examples are specific to the **httpd** service. The command:

```
service httpd graceful
```

is replaced by

```
apachectl graceful
```

The command:

```
service httpd configtest
```

is replaced by

```
apachectl configtest
```

The **systemd** unit file for **httpd** has different behavior from the init script as follows:

- ▶ A graceful restart is used by default when the service is reloaded.
- ▶ A graceful stop is used by default when the service is stopped.

Private /tmp

To enhance system security, the **systemd** unit file runs the **httpd** daemon using a private **/tmp** directory, separate to the system **/tmp** directory.

Configuration Layout

Configuration files which load modules are now placed in the `/etc/httpd/conf.modules.d` directory. Packages, such as `php`, which provide additional loadable modules for `httpd` will place a file in this directory. Any configuration files in the `conf.modules.d` are processed before the main body of `httpd.conf`. Configuration files in the `/etc/httpd/conf.d` directory are now processed after the main body of `httpd.conf`.

Some additional configuration files are provided by the `httpd` package itself:

▶ `/etc/httpd/conf.d/autoindex.conf`

This configures mod_autoindex directory indexing.

▶ `/etc/httpd/conf.d/userdir.conf`

This configures access to user directories, for example, `http://example.com/~username/`; such access is disabled by default for security reasons.

▶ `/etc/httpd/conf.d/welcome.conf`

As in previous releases, this configures the welcome page displayed for `http://localhost/` when no content is present.

Default Configuration

A minimal default `httpd.conf` is now provided by default. Many common configuration settings, such as `Timeout` or `KeepAlive` are no longer explicitly configured in the default configuration; hard-coded settings will be used instead, by default. The hard-coded default settings for all configuration directives are specified in the manual. See [Section 9.1.8.1, “Installed Documentation”](#) for more information.

Configuration Changes

A number of backwards-incompatible changes to the `httpd` configuration syntax were made which will require changes if migrating an existing configuration from `httpd 2.2` to `httpd 2.4`. See the following Apache document for more information on upgrading <http://httpd.apache.org/docs/2.4/upgrading.html>

Processing Model

In previous releases of Red Hat Enterprise Linux, different *multi-processing models* (MPM) were made available as different `httpd` binaries: the forked model, “prefork”, as `/usr/sbin/httpd`, and the thread-based model “worker” as `/usr/sbin/httpd.worker`.

In Red Hat Enterprise Linux 7, only a single `httpd` binary is used, and three MPMs are available as loadable modules: worker, prefork (default), and event. The configuration file `/etc/httpd/conf.modules.d/00-mpm.conf` can be changed to select which of the three MPM modules is loaded.

Packaging Changes

The LDAP authentication and authorization modules are now provided in a separate sub-package `mod_ldap`. The new module `mod_session` and associated helper modules are provided in a new sub-package, `mod_session`. The new modules `mod_proxy_html` and `mod_xml2enc` are provided in a new sub-package `mod_proxv_html`.

provided in a new sub package, `mod_proxy_num`.

Packaging Filesystem Layout

The `/var/cache/mod_proxy` directory is no longer provided; instead, the `/var/cache/httpd/` directory is packaged with a `proxy` and `ssl` subdirectory.

Packaged content provided with `httpd` has been moved from `/var/www/` to `/usr/share/httpd/`:

- ▶ `/usr/share/httpd/icons/`

The `/var/www/icons/` has moved to `/usr/share/httpd/icons`. This directory contains a set of icons used with directory indices. Available at `http://localhost/icons/` in the default configuration, via `/etc/httpd/conf.d/autoindex.conf`.

- ▶ `/usr/share/httpd/manual/`

The `/var/www/manual/` has moved to `/usr/share/httpd/manual/`. This directory, contained in the `httpd-manual` package, contains the HTML version of the manual for `httpd`. Available at `http://localhost/manual/` if the package is installed, via `/etc/httpd/conf.d/manual.conf`.

- ▶ `/usr/share/httpd/error/`

The `/var/www/error/` has moved to `/usr/share/httpd/error/`. Custom multi-language HTTP error pages. Not configured by default, the example configuration file is provided at `/usr/share/doc/httpd-VERSION/httpd-multilang-errordoc.conf`.

Authentication, Authorization and Access Control

The configuration directives used to control authentication, authorization and access control have changed significantly. Existing configuration files using the `Order`, `Deny` and `Allow` directives should be adapted to use the new `Require` syntax. See the following Apache document for more information <http://httpd.apache.org/docs/2.4/howto/auth.html>

`suexec`

To improve system security, the `suexec` binary is no longer installed `setuid root`; instead, it has file system capability bits set which allow a more restrictive set of permissions. In conjunction with this change, the `suexec` binary no longer uses the `/var/log/httpd/suexec.log` logfile. Instead, log messages are sent to `syslog`; by default these will appear in the `/var/log/secure` log file.

Module Interface

Due to changes to the `httpd` module interface, `httpd 2.4` is not compatible with third-party binary modules built against `httpd 2.2`. Such modules will need to be adjusted as necessary for the `httpd 2.4` module interface, and then rebuilt. A detailed list of the API changes in version `2.4` is available here: http://httpd.apache.org/docs/2.4/developer/new_api_2_4.html.

The `apxs` binary used to build modules from source has moved from `/usr/sbin/apxs` to `/usr/bin/apxs`.

Removed modules

List of **httpd** modules removed in Red Hat Enterprise Linux 7:

mod_auth_mysql, **mod_auth_pgsql**

httpd 2.4 provides SQL database authentication support internally in the **mod_authn_dbd** module.

mod_perl

mod_perl is not officially supported with **httpd 2.4** by upstream.

mod_authz_ldap

httpd 2.4 provides LDAP support internally using **mod_authnz_ldap**.

9.1.2. Updating the Configuration

To update the configuration files from the Apache HTTP Server version 2.2, take the following steps:

1. Make sure all module names are correct, since they may have changed. Adjust the **LoadModule** directive for each module that has been renamed.
2. Recompile all third party modules before attempting to load them. This typically means authentication and authorization modules.
3. If you use the **mod_userdir** module, make sure the **UserDir** directive indicating a directory name (typically **public_html**) is provided.
4. If you use the Apache HTTP Secure Server, edit the **/etc/httpd/conf.d/ssl.conf** to enable the Secure Sockets Layer (SSL) protocol.

Note that you can check the configuration for possible errors by using the following command:

```
~]# service httpd configtest  
Syntax OK
```

For more information on upgrading the Apache HTTP Server configuration from version 2.2 to 2.4, see <http://httpd.apache.org/docs/2.4/upgrading.html>.

9.1.3. Running the httpd Service

This section describes how to start, stop, restart, and check the current status of the Apache HTTP Server. To be able to use the **httpd** service, make sure you have the *httpd* installed. You can do so by using the following command:

```
~]# yum install httpd
```

For more information on the concept of targets and how to manage system services in Red Hat Enterprise Linux in general, see [Chapter 6, Managing Services with systemd](#).

9.1.3.1. Starting the Service

To run the **httpd** service, type the following at a shell prompt:

```
~]# systemctl start httpd.service
```

If you want the service to start automatically at the boot time, use the following command:

```
~]# systemctl enable httpd.service
ln -s '/usr/lib/systemd/system/httpd.service' '/etc/systemd/system/multi-user.target.wants/httpd.service'
```



Using the secure server

If running the Apache HTTP Server as a secure server, a password may be required after the machine boots if using an encrypted private SSL key.

9.1.3.2. Stopping the Service

To stop the running **httpd** service, type the following at a shell prompt:

```
~]# systemctl stop httpd.service
```

To prevent the service from starting automatically at the boot time, type:

```
~]# systemctl disable httpd.service
rm '/etc/systemd/system/multi-user.target.wants/httpd.service'
```

9.1.3.3. Restarting the Service

There are three different ways how to restart a running **httpd** service:

1. To restart the service completely, type:

```
~]# systemctl restart httpd.service
```

This stops the running **httpd** service and immediately starts it again. Use this command after installing or removing a dynamically loaded module such as PHP.

2. To only reload the configuration, type:

```
~]# systemctl reload httpd.service
```

This causes the running **httpd** service to reload its configuration file. Any requests being currently processed will be interrupted, which may cause a client browser to display an error message or render a partial page.

3. To reload the configuration without affecting active requests, type:

```
~]# service httpd graceful
```

This cause the running **httpd** service to reload its configuration file. Any requests being currently processed will use the old configuration.

For more information on how to manage system services in Red Hat Enterprise Linux 7, see [Chapter 6, Managing Services with systemd](#).

9.1.3.4. Verifying the Service Status

To verify that the **httpd** service is running, type the following at a shell prompt:

```
~]# systemctl is-active httpd.service
active
```

9.1.4. Editing the Configuration Files

When the **httpd** service is started, by default, it reads the configuration from locations that are listed in [Table 9.1, “The httpd service configuration files”](#).

Table 9.1. The httpd service configuration files

Path	Description
/etc/httpd/conf/httpd.conf	The main configuration file.
/etc/httpd/conf.d/	An auxiliary directory for configuration files that are included in the main configuration file.

Although the default configuration should be suitable for most situations, it is a good idea to become at least familiar with some of the more important configuration options. Note that for any changes to take effect, the web server has to be restarted first. See [Section 9.1.3.3, “Restarting the Service”](#) for more information on how to restart the **httpd** service.

To check the configuration for possible errors, type the following at a shell prompt:

```
~]# service httpd configtest
Syntax OK
```

To make the recovery from mistakes easier, it is recommended that you make a copy of the original file before editing it.

9.1.5. Working with Modules

Being a modular application, the **httpd** service is distributed along with a number of *Dynamic Shared Objects* (DSOs), which can be dynamically loaded or unloaded at runtime as necessary. By default, these modules are located in **/usr/lib/httpd/modules/** on 32-bit and in **/usr/lib64/httpd/modules/** on 64-bit systems.

9.1.5.1. Loading a Module

To load a particular DSO module, use the **LoadModule** directive. Note that modules provided by a separate package often have their own configuration file in the **/etc/httpd/conf.d/** directory.

Example 9.1. Loading the mod_ssl DSO

```
LoadModule ssl_module modules/mod_ssl.so
```

Once you are finished, restart the web server to reload the configuration. See [Section 9.1.3.3, “Restarting the Service”](#) for more information on how to restart the **httpd** service.

9.1.5.2. Writing a Module

If you intend to create a new DSO module, make sure you have the *httpd-devel* package installed. To do so, type the following at a shell prompt:

```
~]# yum install httpd-devel
```

This package contains the include files, the header files, and the **APache eXtenSion (apxs)** utility required to compile a module.

Once written, you can build the module with the following command:

```
~]# apxs -i -a -c module_name.c
```

If the build was successful, you should be able to load the module the same way as any other module that is distributed with the Apache HTTP Server.

9.1.6. Setting Up Virtual Hosts

The Apache HTTP Server's built in virtual hosting allows the server to provide different information based on which IP address, hostname, or port is being requested.

To create a name-based virtual host, find the virtual host container provided in `/etc/httpd/conf/httpd.conf` as an example, remove the hash sign (that is, `#`) from the beginning of each line, and customize the options according to your requirements as shown in [Example 9.2, “Sample virtual host configuration”](#).

Example 9.2. Sample virtual host configuration

```
NameVirtualHost penguin.example.com:80

<VirtualHost penguin.example.com:80>
    ServerAdmin webmaster@penguin.example.com
    DocumentRoot /www/docs/penguin.example.com
    ServerName penguin.example.com:80
    ErrorLog logs/penguin.example.com-error_log
    CustomLog logs/penguin.example.com-access_log common
</VirtualHost>
```

Note that **ServerName** must be a valid DNS name assigned to the machine. The **<VirtualHost>** container is highly customizable, and accepts most of the directives available within the main server configuration. Directives that are *not* supported within this container include **User** and **Group**, which were replaced by **SuexecUserGroup**.

Changing the port number

If you configure a virtual host to listen on a non-default port, make sure you update the **Listen** directive in the global settings section of the `/etc/httpd/conf/httpd.conf` file accordingly.

To activate a newly created virtual host, the web server has to be restarted first. See [Section 9.1.3.3, “Restarting the Service”](#) for more information on how to restart the **httpd** service.

9.1.7. Setting Up an SSL Server

Secure Sockets Layer (SSL) is a cryptographic protocol that allows a server and a client to communicate securely. Along with its extended and improved version called *Transport Layer Security (TLS)*, it ensures both privacy and data integrity. The Apache HTTP Server in combination with **mod_ssl**, a module that uses the OpenSSL toolkit to provide the SSL/TLS support, is commonly referred to as the *SSL server*.

Unlike a regular HTTP connection that can be read and possibly modified by anybody who is able to intercept it, the use of **mod_ssl** prevents any inspection or modification of the transmitted content. This section provides basic information on how to enable this module in the Apache HTTP Server configuration, and guides you through the process of generating private keys and self-signed certificates.

9.1.7.1. An Overview of Certificates and Security

Secure communication is based on the use of keys. In conventional or *symmetric cryptography*, both ends of the transaction have the same key they can use to decode each other's transmissions. On the other hand, in public or *asymmetric cryptography*, two keys co-exist: a *private key* that is kept a secret, and a *public key* that is usually shared with the public. While the data encoded with the public key can only be decoded with the private key, data encoded with the private key can in turn only be decoded with the public key.

To provide secure communications using SSL, an SSL server must use a digital certificate signed by a *Certificate Authority* (CA). The certificate lists various attributes of the server (that is, the server hostname, the name of the company, its location, etc.), and the signature produced using the CA's private key. This signature ensures that a particular certificate authority has issued the certificate, and that the certificate has not been modified in any way.

When a web browser establishes a new SSL connection, it checks the certificate provided by the web server. If the certificate does not have a signature from a trusted CA, or if the hostname listed in the certificate does not match the hostname used to establish the connection, it refuses to communicate with the server and usually presents a user with an appropriate error message.

By default, most web browsers are configured to trust a set of widely used certificate authorities. Because of this, an appropriate CA should be chosen when setting up a secure server, so that target users can trust the connection, otherwise they will be presented with an error message, and will have to accept the certificate manually. Since encouraging users to override certificate errors can allow an attacker to intercept the connection, you should use a trusted CA whenever possible. For more information on this, see [Table 9.2, “CA lists for most common web browsers”](#).

Table 9.2. CA lists for most common web browsers

Web Browser	Link
Mozilla Firefox	Mozilla root CA list .
Opera	Root certificates used by Opera .
Internet Explorer	Windows root certificate program members .

When setting up an SSL server, you need to generate a certificate request and a private key, and then send the certificate request, proof of the company's identity, and payment to a certificate authority. Once the CA verifies the certificate request and your identity, it will send you a signed certificate you can use with your server. Alternatively, you can create a self-signed certificate that does not contain a CA signature, and thus should be used for testing purposes only.

9.1.7.2. Enabling the mod_ssl Module

If you intend to set up an SSL server, make sure you have the *mod_ssl* (the **mod_ssl** module) and *openssl* (the OpenSSL toolkit) packages installed. To do so, type the following at a shell prompt:

```
~]# yum install mod_ssl openssl
```

This will create the **mod_ssl** configuration file at **/etc/httpd/conf.d/ssl.conf**, which is included in the main Apache HTTP Server configuration file by default. For the module to be loaded, restart the **httpd** service as described in [Section 9.1.3.3, “Restarting the Service”](#).

9.1.7.3. Using an Existing Key and Certificate

If you have a previously created key and certificate, you can configure the SSL server to use these files instead of generating new ones. There are only two situations where this is not possible:

1. You are changing the IP address or domain name.

Certificates are issued for a particular IP address and domain name pair. If one of these values changes, the certificate becomes invalid.

2. You have a certificate from VeriSign, and you are changing the server software.

VeriSign, a widely used certificate authority, issues certificates for a particular software product, IP address, and domain name. Changing the software product renders the certificate invalid.

In either of the above cases, you will need to obtain a new certificate. For more information on this topic, see [Section 9.1.7.4, “Generating a New Key and Certificate”](#).

If you wish to use an existing key and certificate, move the relevant files to the `/etc/pki/tls/private/` and `/etc/pki/tls/certs/` directories respectively. You can do so by typing the following commands:

```
~]# mv key_file.key /etc/pki/tls/private/hostname.key
~]# mv certificate.crt /etc/pki/tls/certs/hostname.crt
```

Then add the following lines to the `/etc/httpd/conf.d/ssl.conf` configuration file:

```
SSLCertificateFile /etc/pki/tls/certs/hostname.crt
SSLCertificateKeyFile /etc/pki/tls/private/hostname.key
```

To load the updated configuration, restart the `httpd` service as described in [Section 9.1.3.3, “Restarting the Service”](#).

Example 9.3. Using a key and certificate from the Red Hat Secure Web Server

```
~]# mv /etc/httpd/conf/httpsd.key /etc/pki/tls/private/penguin.example.com.key
~]# mv /etc/httpd/conf/httpsd.crt /etc/pki/tls/certs/penguin.example.com.crt
```

9.1.7.4. Generating a New Key and Certificate

In order to generate a new key and certificate pair, you must have the `crypto-utils` package installed in your system. You can install it by typing the following at a shell prompt:

```
~]# yum install crypto-utils
```

This package provides a set of tools to generate and manage SSL certificates and private keys, and includes `genkey`, the Red Hat Keypair Generation utility that will guide you through the key generation process.



Replacing an existing certificate

If the server already has a valid certificate and you are replacing it with a new one, specify a different serial number. This ensures that client browsers are notified of this change, update to this new certificate as expected, and do not fail to access the page. To create a new certificate with a custom serial number, use the following command instead of **genkey**:

```
~]# openssl req -x509 -new -set_serial number -key hostname.key -out hostname.crt
```



Remove a previously created key

If there already is a key file for a particular hostname in your system, **genkey** will refuse to start. In this case, remove the existing file using the following command:

```
~]# rm /etc/pki/tls/private/hostname.key
```

To run the utility, use the **genkey** command followed by the appropriate hostname (for example, **penguin.example.com**):

```
~]# genkey hostname
```

To complete the key and certificate creation, take the following steps:

1. Review the target locations in which the key and certificate will be stored.

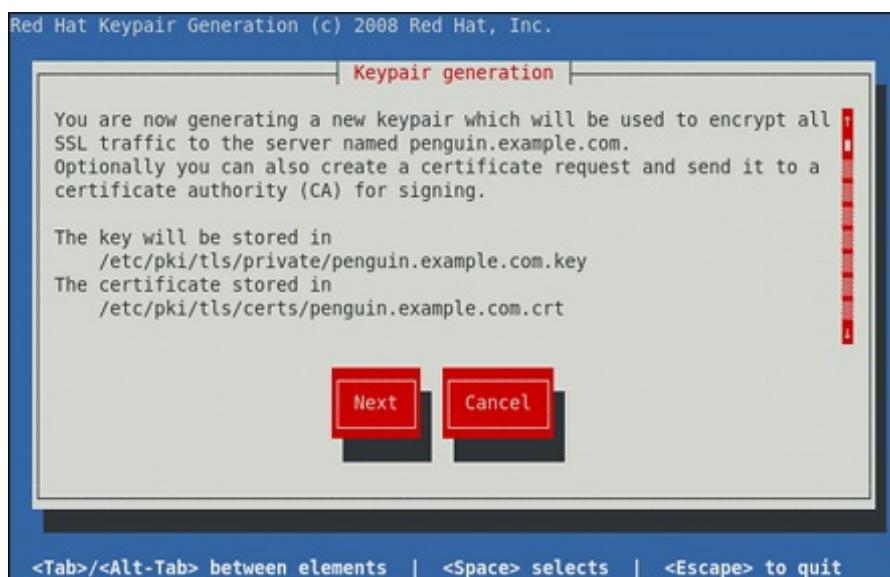


Figure 9.1. Running the genkey utility

Use the **Tab** key to select the **Next** button, and press **Enter** to proceed to the next screen.

- Using the **Up** and **down** arrow keys, select the suitable key size. Note that while the large key increases the security, it also increases the response time of your server. The NIST recommends using **2048 bits**. See [NIST Special Publication 800-131A](#).

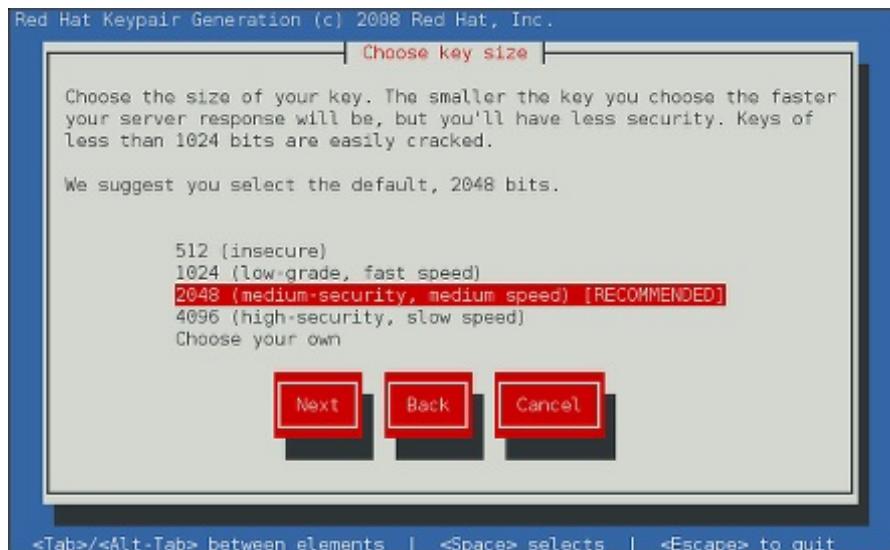


Figure 9.2. Selecting the key size

Once finished, use the **Tab** key to select the **Next** button, and press **Enter** to initiate the random bits generation process. Depending on the selected key size, this may take some time.

- Decide whether you wish to send a certificate request to a certificate authority.



Figure 9.3. Generating a certificate request

Use the **Tab** key to select **Yes** to compose a certificate request, or **No** to generate a self-signed certificate. Then press **Enter** to confirm your choice.

- Using the **Spacebar** key, enable ([*]) or disable ([]) the encryption of the private key.

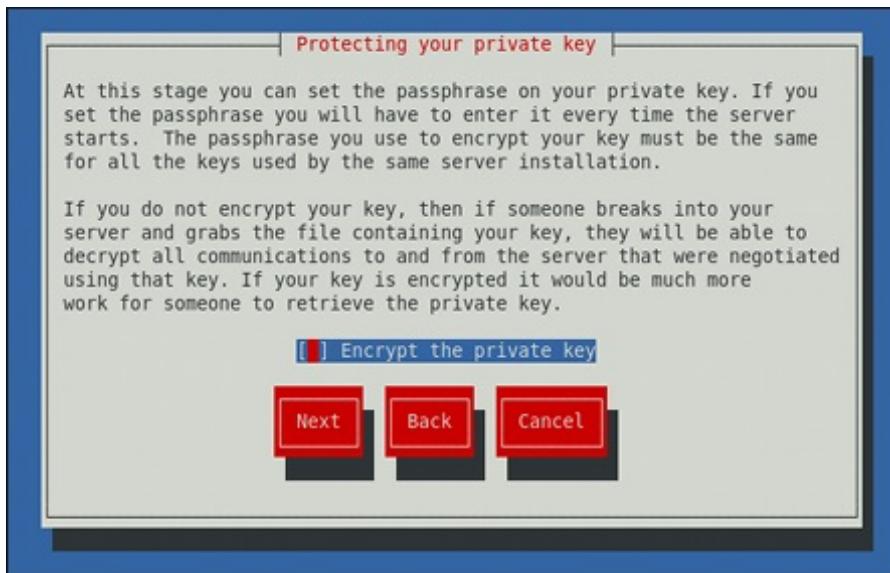


Figure 9.4. Encrypting the private key

Use the **Tab** key to select the **Next** button, and press **Enter** to proceed to the next screen.

5. If you have enabled the private key encryption, enter an adequate passphrase. Note that for security reasons, it is not displayed as you type, and it must be at least five characters long.

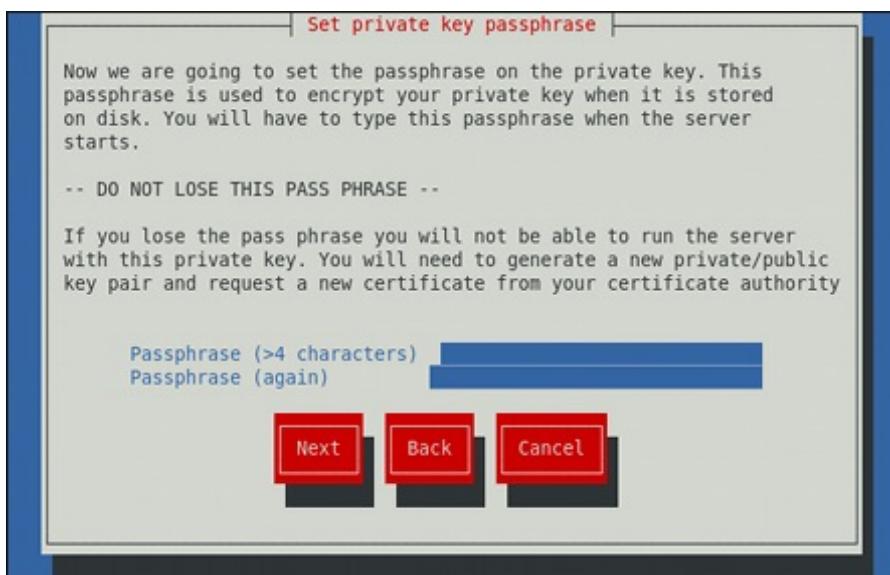
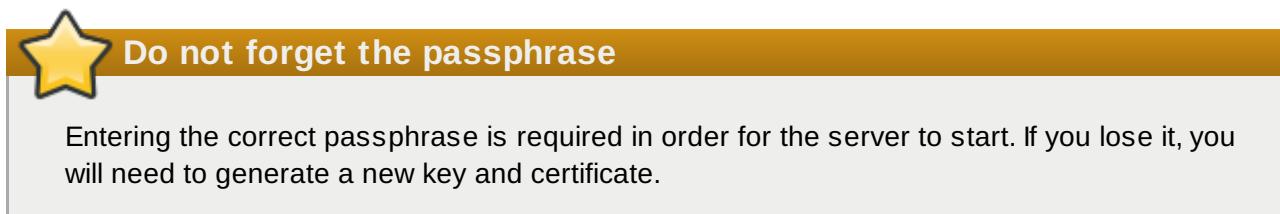


Figure 9.5. Entering a passphrase

Use the **Tab** key to select the **Next** button, and press **Enter** to proceed to the next screen.



6. Customize the certificate details.



Figure 9.6. Specifying certificate information

Use the **Tab** key to select the **Next** button, and press **Enter** to finish the key generation.

7. If you have previously enabled the certificate request generation, you will be prompted to send it to a certificate authority.

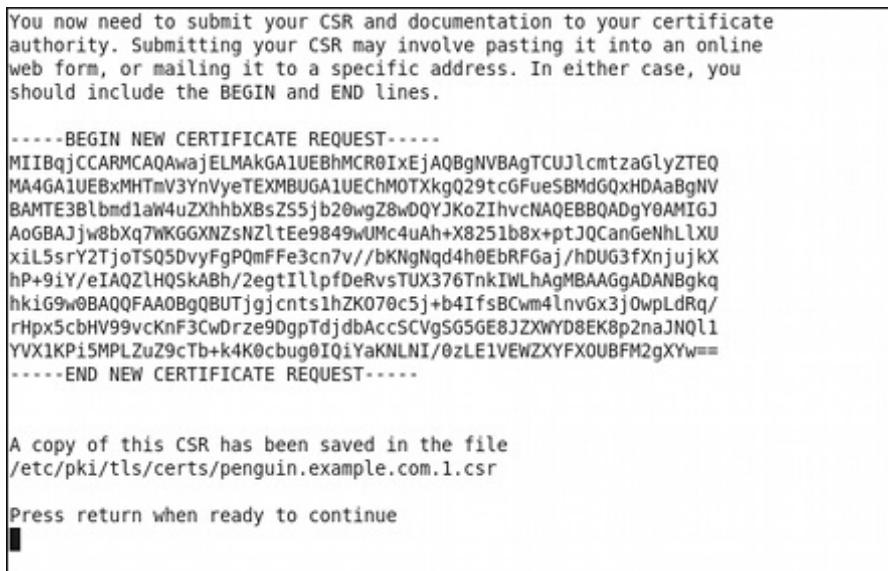


Figure 9.7. Instructions on how to send a certificate request

Press **Enter** to return to a shell prompt.

Once generated, add the key and certificate locations to the `/etc/httpd/conf.d/ssl.conf` configuration file:

```
SSLCertificateFile /etc/pki/tls/certs/hostname.crt
SSLCertificateKeyFile /etc/pki/tls/private/hostname.key
```

Finally, restart the **httpd** service as described in [Section 9.1.3.3, “Restarting the Service”](#), so that the updated configuration is loaded.

9.1.8. Additional Resources

To learn more about the Apache HTTP Server, see the following resources.

9.1.8.1. Installed Documentation

<http://localhost/manual/>

The official documentation for the Apache HTTP Server with the full description of its directives and available modules. Note that in order to access this documentation, you must have the *httpd-manual* package installed, and the web server must be running.

man httpd

The manual page for the **httpd** service containing the complete list of its command line options.

man genkey

The manual page for **genkey** containing the full documentation on its usage.

9.1.8.2. Useful Websites

<http://httpd.apache.org/>

The official website for the Apache HTTP Server with documentation on all the directives and default modules.

<http://www.modssl.org/>

The official website for the **mod_ssl** module.

<http://www.openssl.org/>

The OpenSSL home page containing further documentation, frequently asked questions, links to the mailing lists, and other useful resources.

Chapter 10. Mail Servers

Red Hat Enterprise Linux offers many advanced applications to serve and access email. This chapter describes modern email protocols in use today, and some of the programs designed to send and receive email.

10.1. Email Protocols

Today, email is delivered using a client/server architecture. An email message is created using a mail client program. This program then sends the message to a server. The server then forwards the message to the recipient's email server, where the message is then supplied to the recipient's email client.

To enable this process, a variety of standard network protocols allow different machines, often running different operating systems and using different email programs, to send and receive email.

The following protocols discussed are the most commonly used in the transfer of email.

10.1.1. Mail Transport Protocols

Mail delivery from a client application to the server, and from an originating server to the destination server, is handled by the *Simple Mail Transfer Protocol (SMTP)*.

10.1.1.1. SMTP

The primary purpose of SMTP is to transfer email between mail servers. However, it is critical for email clients as well. To send email, the client sends the message to an outgoing mail server, which in turn contacts the destination mail server for delivery. For this reason, it is necessary to specify an SMTP server when configuring an email client.

Under Red Hat Enterprise Linux, a user can configure an SMTP server on the local machine to handle mail delivery. However, it is also possible to configure remote SMTP servers for outgoing mail.

One important point to make about the SMTP protocol is that it does not require authentication. This allows anyone on the Internet to send email to anyone else or even to large groups of people. It is this characteristic of SMTP that makes junk email or *spam* possible. Imposing relay restrictions limits random users on the Internet from sending email through your SMTP server, to other servers on the internet. Servers that do not impose such restrictions are called *open relay* servers.

Red Hat Enterprise Linux 7 provides the Postfix and Sendmail SMTP programs.

10.1.2. Mail Access Protocols

There are two primary protocols used by email client applications to retrieve email from mail servers: the *Post Office Protocol (POP)* and the *Internet Message Access Protocol (IMAP)*.

10.1.2.1. POP

The default POP server under Red Hat Enterprise Linux is **Dovecot** and is provided by the *dovecot* package.

**Note**

In order to use **Dovecot**, first ensure the **dovecot** package is installed on your system by running, as **root**:

```
~]# yum install dovecot
```

For more information on installing packages with Yum, see [Section 5.2.4, “Installing Packages”](#).

When using a **POP** server, email messages are downloaded by email client applications. By default, most **POP** email clients are automatically configured to delete the message on the email server after it has been successfully transferred, however this setting usually can be changed.

POP is fully compatible with important Internet messaging standards, such as *Multipurpose Internet Mail Extensions (MIME)*, which allow for email attachments.

POP works best for users who have one system on which to read email. It also works well for users who do not have a persistent connection to the Internet or the network containing the mail server. Unfortunately for those with slow network connections, **POP** requires client programs upon authentication to download the entire content of each message. This can take a long time if any messages have large attachments.

The most current version of the standard **POP** protocol is **POP3**.

There are, however, a variety of lesser-used **POP** protocol variants:

- ▶ **APOP** — **POP3** with **MD5** authentication. An encoded hash of the user's password is sent from the email client to the server rather than sending an unencrypted password.
- ▶ **KPOP** — **POP3** with Kerberos authentication.
- ▶ **RPOP** — **POP3** with **RPOP** authentication. This uses a per-user ID, similar to a password, to authenticate POP requests. However, this ID is not encrypted, so **RPOP** is no more secure than standard **POP**.

For added security, it is possible to use *Secure Socket Layer (SSL)* encryption for client authentication and data transfer sessions. This can be enabled by using the **pop3s** service, or by using the **stunnel** application. For more information on securing email communication, see [Section 10.5.1, “Securing Communication”](#).

10.1.2.2. IMAP

The default **IMAP** server under Red Hat Enterprise Linux is **Dovecot** and is provided by the **dovecot** package. See [Section 10.1.2.1, “POP”](#) for information on how to install **Dovecot**.

When using an **IMAP** mail server, email messages remain on the server where users can read or delete them. **IMAP** also allows client applications to create, rename, or delete mail directories on the server to organize and store email.

IMAP is particularly useful for users who access their email using multiple machines. The protocol is also convenient for users connecting to the mail server via a slow connection, because only the email header information is downloaded for messages until opened, saving bandwidth. The user also has the ability to delete messages without viewing or downloading them.

For convenience, **IMAP** client applications are capable of caching copies of messages locally, so the user

can browse previously read messages when not directly connected to the **IMAP** server.

IMAP, like **POP**, is fully compatible with important Internet messaging standards, such as **MIME**, which allow for email attachments.

For added security, it is possible to use **SSL** encryption for client authentication and data transfer sessions. This can be enabled by using the **imaps** service, or by using the **stunnel** program. For more information on securing email communication, see [Section 10.5.1, “Securing Communication”](#).

Other free, as well as commercial, IMAP clients and servers are available, many of which extend the IMAP protocol and provide additional functionality.

10.1.2.3. Dovecot

The **imap-login** and **pop3-login** processes which implement the **IMAP** and **POP3** protocols are spawned by the master **dovecot** daemon included in the *dovecot* package. The use of **IMAP** and **POP** is configured through the **/etc/dovecot/dovecot.conf** configuration file; by default **dovecot** runs **IMAP** and **POP3** together with their secure versions using **SSL**. To configure **dovecot** to use **POP**, complete the following steps:

1. Edit the **/etc/dovecot/dovecot.conf** configuration file to make sure the **protocols** variable is uncommented (remove the hash sign (#) at the beginning of the line) and contains the **pop3** argument. For example:

```
protocols = imap imaps pop3 pop3s
```

When the **protocols** variable is left commented out, **dovecot** will use the default values will use the default values as described above.

2. Make the change operational for the current session by running the following command:

```
~]# systemctl restart dovecot
```

3. Make the change operational after the next reboot by running the command:

```
~]# systemctl enable dovecot
ln -s '/usr/lib/systemd/system/dovecot' '/etc/systemd/system/multi-
user.target.wants/dovecot'
```



Note

Please note that **dovecot** only reports that it started the **IMAP** server, but also starts the **POP3** server.

Unlike **SMTP**, both **IMAP** and **POP3** require connecting clients to authenticate using a username and password. By default, passwords for both protocols are passed over the network unencrypted.

To configure **SSL** on **dovecot**:

- ▶ Edit the **/etc/pki/dovecot/dovecot-openssl.cnf** configuration file as you prefer. However, in a typical installation, this file does not require modification.
- ▶ Rename, move or delete the files **/etc/pki/dovecot/certs/dovecot.pem** and **/etc/pki/dovecot/private/dovecot.pem**.

- ▶ Execute the `/usr/libexec/dovecot/mkcert.sh` script which creates the **dovecot** self signed certificates. These certificates are copied in the `/etc/pki/dovecot/certs` and `/etc/pki/dovecot/private` directories. To implement the changes, restart **dovecot**:

```
~]# systemctl restart dovecot
```

More details on **dovecot** can be found online at <http://www.dovecot.org>.

10.2. Email Program Classifications

In general, all email applications fall into at least one of three classifications. Each classification plays a specific role in the process of moving and managing email messages. While most users are only aware of the specific email program they use to receive and send messages, each one is important for ensuring that email arrives at the correct destination.

10.2.1. Mail Transport Agent

A *Mail Transport Agent (MTA)* transports email messages between hosts using **SMTP**. A message may involve several MTAs as it moves to its intended destination.

While the delivery of messages between machines may seem rather straightforward, the entire process of deciding if a particular MTA can or should accept a message for delivery is quite complicated. In addition, due to problems from spam, use of a particular MTA is usually restricted by the MTA's configuration or the access configuration for the network on which the MTA resides.

Many modern email client programs can act as an MTA when sending email. However, this action should not be confused with the role of a true MTA. The sole reason email client programs are capable of sending email like an MTA is because the host running the application does not have its own MTA. This is particularly true for email client programs on non-UNIX-based operating systems. However, these client programs only send outbound messages to an MTA they are authorized to use and do not directly deliver the message to the intended recipient's email server.

Since Red Hat Enterprise Linux offers two MTAs, *Postfix* and *Sendmail*, email client programs are often not required to act as an MTA. Red Hat Enterprise Linux also includes a special purpose MTA called *Fetchmail*.

For more information on Postfix, Sendmail, and Fetchmail, see [Section 10.3, “Mail Transport Agents”](#).

10.2.2. Mail Delivery Agent

A *Mail Delivery Agent (MDA)* is invoked by the MTA to file incoming email in the proper user's mailbox. In many cases, the MDA is actually a *Local Delivery Agent (LDA)*, such as **mail** or **Procmail**.

Any program that actually handles a message for delivery to the point where it can be read by an email client application can be considered an MDA. For this reason, some MTAs (such as *Sendmail* and *Postfix*) can fill the role of an MDA when they append new email messages to a local user's mail spool file. In general, MDAs do not transport messages between systems nor do they provide a user interface; MDAs distribute and sort messages on the local machine for an email client application to access.

10.2.3. Mail User Agent

A *Mail User Agent (MUA)* is synonymous with an email client application. An MUA is a program that, at a minimum, allows a user to read and compose email messages. Many MUAs are capable of retrieving messages via the **POP** or **IMAP** protocols, setting up mailboxes to store messages, and sending outbound messages to an MTA.

MUAs may be graphical, such as **Evolution**, or have simple text-based interfaces, such as **Mutt**.

10.3. Mail Transport Agents

Red Hat Enterprise Linux 7 offers two primary MTAs: Postfix and Sendmail. Postfix is configured as the default MTA and Sendmail is considered deprecated. If required to switch the default MTA to Sendmail, you can either uninstall Postfix or use the following command to switch to Sendmail:

```
~]# alternatives --config mta
```

You can also use the following command to enable the desired service:

```
~]# systemctl enable <service>
```

Similarly, to disable the service, type the following at a shell prompt:

```
~]# systemctl disable <service>
```

For more information on how to manage system services in Red Hat Enterprise Linux 7, see [Chapter 6, Managing Services with systemd](#).

10.3.1. Postfix

Originally developed at IBM by security expert and programmer Wietse Venema, Postfix is a Sendmail-compatible MTA that is designed to be secure, fast, and easy to configure.

To improve security, Postfix uses a modular design, where small processes with limited privileges are launched by a *master* daemon. The smaller, less privileged processes perform very specific tasks related to the various stages of mail delivery and run in a chrooted environment to limit the effects of attacks.

Configuring Postfix to accept network connections from hosts other than the local computer takes only a few minor changes in its configuration file. Yet for those with more complex needs, Postfix provides a variety of configuration options, as well as third party add-ons that make it a very versatile and full-featured MTA.

The configuration files for Postfix are human readable and support up to 250 directives. Unlike Sendmail, no macro processing is required for changes to take effect and the majority of the most commonly used options are described in the heavily commented files.

10.3.1.1. The Default Postfix Installation

The Postfix executable is **postfix**. This daemon launches all related processes needed to handle mail delivery.

Postfix stores its configuration files in the **/etc/postfix/** directory. The following is a list of the more commonly used files:

- ▶ **access** — Used for access control, this file specifies which hosts are allowed to connect to Postfix.
- ▶ **main.cf** — The global Postfix configuration file. The majority of configuration options are specified in this file.
- ▶ **master.cf** — Specifies how Postfix interacts with various processes to accomplish mail delivery.
- ▶ **transport** — Maps email addresses to relay hosts.

The **aliases** file can be found in the `/etc/` directory. This file is shared between Postfix and Sendmail. It is a configurable list required by the mail protocol that describes user ID aliases.



Important

The default `/etc/postfix/main.cf` file does not allow Postfix to accept network connections from a host other than the local computer. For instructions on configuring Postfix as a server for other clients, see [Section 10.3.1.2, “Basic Postfix Configuration”](#).

Restart the **postfix** service after changing any options in the configuration files under the `/etc/postfix` directory in order for those changes to take effect:

```
~]# systemctl restart postfix
```

10.3.1.2. Basic Postfix Configuration

By default, Postfix does not accept network connections from any host other than the local host. Perform the following steps as **root** to enable mail delivery for other hosts on the network:

- ▶ Edit the `/etc/postfix/main.cf` file with a text editor, such as **vi**.
- ▶ Uncomment the **mydomain** line by removing the hash sign (#), and replace *domain.tld* with the domain the mail server is servicing, such as **example.com**.
- ▶ Uncomment the **myorigin = \$mydomain** line.
- ▶ Uncomment the **myhostname** line, and replace *host.domain.tld* with the hostname for the machine.
- ▶ Uncomment the **mydestination = \$myhostname, localhost.\$mydomain** line.
- ▶ Uncomment the **mynetworks** line, and replace *168.100.189.0/28* with a valid network setting for hosts that can connect to the server.
- ▶ Uncomment the **inet_interfaces = all** line.
- ▶ Comment the **inet_interfaces = localhost** line.
- ▶ Restart the **postfix** service.

Once these steps are complete, the host accepts outside emails for delivery.

Postfix has a large assortment of configuration options. One of the best ways to learn how to configure Postfix is to read the comments within the `/etc/postfix/main.cf` configuration file. Additional resources including information about Postfix configuration, SpamAssassin integration, or detailed descriptions of the `/etc/postfix/main.cf` parameters are available online at <http://www.postfix.org/>.

10.3.1.3. Using Postfix with LDAP

Postfix can use an **LDAP** directory as a source for various lookup tables (e.g.: **aliases**, **virtual**, **canonical**, etc.). This allows **LDAP** to store hierarchical user information and Postfix to only be given the result of **LDAP** queries when needed. By not storing this information locally, administrators can easily maintain it.

10.3.1.3.1. The `/etc/aliases` lookup example

The following is a basic example for using **LDAP** to look up the **/etc/aliases** file. Make sure your **/etc/postfix/main.cf** file contains the following:

```
alias_maps = hash:/etc/aliases, ldap:/etc/postfix/ldap-aliases.cf
```

Create a **/etc/postfix/ldap-aliases.cf** file if you do not have one already and make sure it contains the following:

```
server_host = ldap.example.com
search_base = dc=example, dc=com
```

where **ldap.example.com**, **example**, and **com** are parameters that need to be replaced with specification of an existing available **LDAP** server.

Note

The **/etc/postfix/ldap-aliases.cf** file can specify various parameters, including parameters that enable **LDAP SSL** and **STARTTLS**. For more information, see the **ldap_table(5)** man page.

For more information on **LDAP**, see [Section 11.1, “OpenLDAP”](#).

10.3.2. Sendmail

Sendmail's core purpose, like other MTAs, is to safely transfer email among hosts, usually using the **SMTP** protocol. Note that Sendmail is considered deprecated and users are encouraged to use Postfix when possible. See [Section 10.3.1, “Postfix”](#) for more information.

10.3.2.1. Purpose and Limitations

It is important to be aware of what Sendmail is and what it can do, as opposed to what it is not. In these days of monolithic applications that fulfill multiple roles, Sendmail may seem like the only application needed to run an email server within an organization. Technically, this is true, as Sendmail can spool mail to each users' directory and deliver outbound mail for users. However, most users actually require much more than simple email delivery. Users usually want to interact with their email using an MUA, that uses **POP** or **IMAP**, to download their messages to their local machine. Or, they may prefer a Web interface to gain access to their mailbox. These other applications can work in conjunction with Sendmail, but they actually exist for different reasons and can operate separately from one another.

It is beyond the scope of this section to go into all that Sendmail should or could be configured to do. With literally hundreds of different options and rule sets, entire volumes have been dedicated to helping explain everything that can be done and how to fix things that go wrong. See the [Section 10.6, “Additional Resources”](#) for a list of Sendmail resources.

This section reviews the files installed with Sendmail by default and reviews basic configuration changes, including how to stop unwanted email (spam) and how to extend Sendmail with the *Lightweight Directory Access Protocol (LDAP)*.

10.3.2.2. The Default Sendmail Installation

In order to use Sendmail, first ensure the *sendmail* package is installed on your system by running, as **root**:

```
~]# yum install sendmail
```

In order to configure Sendmail, ensure the `sendmail-cf` package is installed on your system by running, as **root**:

```
~]# yum install sendmail-cf
```

For more information on installing packages with Yum, see [Section 5.2.4, “Installing Packages”](#).

Before using Sendmail, the default MTA has to be switched from Postfix. For more information how to switch the default MTA refer to [Section 10.3, “Mail Transport Agents”](#).

The Sendmail executable is **sendmail**.

Sendmail's lengthy and detailed configuration file is `/etc/mail/sendmail.cf`. Avoid editing the `sendmail.cf` file directly. To make configuration changes to Sendmail, edit the `/etc/mail/sendmail.mc` file, back up the original `/etc/mail/sendmail.cf` file, and use the following alternatives to generate a new configuration file:

- ▶ Use the included makefile in `/etc/mail/` to create a new `/etc/mail/sendmail.cf` configuration file:

```
~]# make all -C /etc/mail/
```

All other generated files in `/etc/mail` (db files) will be regenerated if needed. The old makemap commands are still usable. The make command is automatically used whenever you start or restart the **sendmail** service.

More information on configuring Sendmail can be found in [Section 10.3.2.3, “Common Sendmail Configuration Changes”](#).

Various Sendmail configuration files are installed in the `/etc/mail/` directory including:

- ▶ **access** — Specifies which systems can use Sendmail for outbound email.
- ▶ **domaintable** — Specifies domain name mapping.
- ▶ **local-host-names** — Specifies aliases for the host.
- ▶ **mailertable** — Specifies instructions that override routing for particular domains.
- ▶ **virtusertable** — Specifies a domain-specific form of aliasing, allowing multiple virtual domains to be hosted on one machine.

Several of the configuration files in `/etc/mail/`, such as **access**, **domaintable**, **mailertable** and **virtusertable**, must actually store their information in database files before Sendmail can use any configuration changes. To include any changes made to these configurations in their database files, run the following commands, as **root**:

```
~]# cd /etc/mail
~]# make all
```

This will update **virtusertable.db**, **access.db**, **domaintable.db**, **mailertable.db**, **sendmail.cf**, and **submit.cf**.

To update all the database files listed above and to update a custom database file, use a command in the following format:

```
make <name>.db all
```

where `<name>` represents the name of the custom database file to be updated.

To update a single database, use a command in the following format:

```
make <name>.db
```

where `<name>.db` represents the name of the database file to be updated.

You may also restart the `sendmail` service for the changes to take effect by running:

```
~]# systemctl restart sendmail
```

For example, to have all emails addressed to the `example.com` domain delivered to `bob@other-example.com`, add the following line to the `virtusertable` file:

```
@example.com bob@other-example.com
```

To finalize the change, the `virtusertable.db` file must be updated:

```
~]# make virtusertable.db all
```

Using the `all` option will result in the `virtusertable.db` and `access.db` being updated at the same time.

10.3.2.3. Common Sendmail Configuration Changes

When altering the Sendmail configuration file, it is best not to edit an existing file, but to generate an entirely new `/etc/mail/sendmail.cf` file.



Warning

Before replacing or making any changes to the `sendmail.cf` file, create a backup copy.

To add the desired functionality to Sendmail, edit the `/etc/mail/sendmail.mc` file as `root`. Once you are finished, restart the `sendmail` service and, if the `m4` package is installed, the `m4` macro processor will automatically generate a new `sendmail.cf` configuration file:

```
~]# systemctl restart sendmail
```



Important

The default `sendmail.cf` file does not allow Sendmail to accept network connections from any host other than the local computer. To configure Sendmail as a server for other clients, edit the `/etc/mail/sendmail.mc` file, and either change the address specified in the `Addr=` option of the `DAEMON_OPTIONS` directive from `127.0.0.1` to the IP address of an active network device or comment out the `DAEMON_OPTIONS` directive all together by placing `dnl` at the beginning of the line. When finished, regenerate `/etc/mail/sendmail.cf` by restarting the service

```
~]# systemctl restart sendmail
```

The default configuration in Red Hat Enterprise Linux works for most **SMTP**-only sites. However, it does not work for *UUCP (UNIX-to-UNIX Copy Protocol)* sites. If using UUCP mail transfers, the `/etc/mail/sendmail.mc` file must be reconfigured and a new `/etc/mail/sendmail.cf` file must be generated.

Consult the `/usr/share/sendmail-cf/README` file before editing any files in the directories under the `/usr/share/sendmail-cf` directory, as they can affect the future configuration of the `/etc/mail/sendmail.cf` file.

10.3.2.4. Masquerading

One common Sendmail configuration is to have a single machine act as a mail gateway for all machines on the network. For example, a company may want to have a machine called `mail.example.com` that handles all of their email and assigns a consistent return address to all outgoing mail.

In this situation, the Sendmail server must masquerade the machine names on the company network so that their return address is `user@example.com` instead of `user@host.example.com`.

To do this, add the following lines to `/etc/mail/sendmail.mc`:

```
FEATURE(always_add_domain)dnl
FEATURE(`masquerade_entire_domain')dnl
FEATURE(`masquerade_envelope')dnl
FEATURE(`allmasquerade')dnl
MASQUERADE_AS(`example.com.')dnl
MASQUERADE_DOMAIN(`example.com.')dnl
MASQUERADE_AS(example.com)dnl
```

After generating a new `sendmail.cf` file using the `m4` macro processor, this configuration makes all mail from inside the network appear as if it were sent from `example.com`.

Note that administrators of mail servers, **DNS** and **DHCP** servers, as well as any provisioning applications, should agree on the host name format used in an organization. See the [Red Hat Enterprise Linux 7 Networking Guide](#) for more information on recommended naming practices.

10.3.2.5. Stopping Spam

Email spam can be defined as unnecessary and unwanted email received by a user who never requested the communication. It is a disruptive, costly, and widespread abuse of Internet communication standards.

Sendmail makes it relatively easy to block new spamming techniques being employed to send junk email. It even blocks many of the more usual spamming methods by default. Main anti-spam features available in sendmail are *header checks*, *relaying denial* (default from version 8.9), *access database and sender information checks*.

For example, forwarding of **SMTP** messages, also called relaying, has been disabled by default since Sendmail version 8.9. Before this change occurred, Sendmail directed the mail host (**x.edu**) to accept messages from one party (**y.com**) and sent them to a different party (**z.net**). Now, however, Sendmail must be configured to permit any domain to relay mail through the server. To configure relay domains, edit the **/etc/mail/relay-domains** file and restart Sendmail

```
~]# systemctl restart sendmail
```

However users can also be sent spam from servers on the Internet. In these instances, Sendmail's access control features available through the **/etc/mail/access** file can be used to prevent connections from unwanted hosts. The following example illustrates how this file can be used to both block and specifically allow access to the Sendmail server:

```
badspammer.com ERROR:550 "Go away and do not spam us anymore"  
tux.badspammer.com OK 10.0 RELAY
```

This example shows that any email sent from **badspammer.com** is blocked with a 550 RFC-821 compliant error code, with a message sent back. Email sent from the **tux.badspammer.com** sub-domain, is accepted. The last line shows that any email sent from the 10.0.*.* network can be relayed through the mail server.

Because the **/etc/mail/access.db** file is a database, use the **makemap** command to update any changes. Do this using the following command as **root**:

```
~]# makemap hash /etc/mail/access < /etc/mail/access
```

Message header analysis allows you to reject mail based on header contents. **SMTP** servers store information about an email's journey in the message header. As the message travels from one MTA to another, each puts in a **Received** header above all the other **Received** headers. It is important to note that this information may be altered by spammers.

The above examples only represent a small part of what Sendmail can do in terms of allowing or blocking access. See the **/usr/share/sendmail-cf/README** file for more information and examples.

Since Sendmail calls the Procmail MDA when delivering mail, it is also possible to use a spam filtering program, such as SpamAssassin, to identify and file spam for users. See [Section 10.4.2.6, “Spam Filters”](#) for more information about using SpamAssassin.

10.3.2.6. Using Sendmail with LDAP

Using **LDAP** is a very quick and powerful way to find specific information about a particular user from a much larger group. For example, an **LDAP** server can be used to look up a particular email address from a common corporate directory by the user's last name. In this kind of implementation, **LDAP** is largely separate from Sendmail, with **LDAP** storing the hierarchical user information and Sendmail only being given the result of **LDAP** queries in pre-addressed email messages.

However, Sendmail supports a much greater integration with **LDAP**, where it uses **LDAP** to replace separately maintained files, such as **/etc/aliases** and **/etc/mail/virtusertables**, on different mail servers that work together to support a medium- to enterprise-level organization. In short, **LDAP** abstracts the mail routing level from Sendmail and its separate configuration files to a powerful **LDAP** cluster that can be leveraged by many different applications.

The current version of Sendmail contains support for **LDAP**. To extend the Sendmail server using **LDAP**, first get an **LDAP** server, such as **OpenLDAP**, running and properly configured. Then edit the **/etc/mail/sendmail.mc** to include the following:

```
LDAPROUTE_DOMAIN('yourdomain.com')dn1
FEATURE('ldap_routing')dn1
```

Note

This is only for a very basic configuration of Sendmail with **LDAP**. The configuration can differ greatly from this depending on the implementation of **LDAP**, especially when configuring several Sendmail machines to use a common **LDAP** server.

Consult **/usr/share/sendmail-cf/README** for detailed **LDAP** routing configuration instructions and examples.

Next, recreate the **/etc/mail/sendmail.cf** file by running the **m4** macro processor and again restarting Sendmail. See [Section 10.3.2.3, “Common Sendmail Configuration Changes”](#) for instructions.

For more information on **LDAP**, see [Section 11.1, “OpenLDAP”](#).

10.3.3. Fetchmail

Fetchmail is an MTA which retrieves email from remote servers and delivers it to the local MTA. Many users appreciate the ability to separate the process of downloading their messages located on a remote server from the process of reading and organizing their email in an MUA. Designed with the needs of dial-up users in mind, Fetchmail connects and quickly downloads all of the email messages to the mail spool file using any number of protocols, including **POP3** and **IMAP**. It can even forward email messages to an **SMTP** server, if necessary.

Note

In order to use **Fetchmail**, first ensure the *fetchmail* package is installed on your system by running, as **root**:

```
~]# yum install fetchmail
```

For more information on installing packages with Yum, see [Section 5.2.4, “Installing Packages”](#).

Fetchmail is configured for each user through the use of a **.fetchmailrc** file in the user's home directory. If it does not already exist, create the **.fetchmailrc** file in your home directory

Using preferences in the **.fetchmailrc** file, Fetchmail checks for email on a remote server and downloads it. It then delivers it to port **25** on the local machine, using the local MTA to place the email in

the correct user's spool file. If Procmail is available, it is launched to filter the email and place it in a mailbox so that it can be read by an MUA.

10.3.3.1. Fetchmail Configuration Options

Although it is possible to pass all necessary options on the command line to check for email on a remote server when executing Fetchmail, using a `.fetchmailrc` file is much easier. Place any desired configuration options in the `.fetchmailrc` file for those options to be used each time the `fetchmail` command is issued. It is possible to override these at the time Fetchmail is run by specifying that option on the command line.

A user's `.fetchmailrc` file contains three classes of configuration options:

- ▶ *global options* — Gives Fetchmail instructions that control the operation of the program or provide settings for every connection that checks for email.
- ▶ *server options* — Specifies necessary information about the server being polled, such as the hostname, as well as preferences for specific email servers, such as the port to check or number of seconds to wait before timing out. These options affect every user using that server.
- ▶ *user options* — Contains information, such as username and password, necessary to authenticate and check for email using a specified email server.

Global options appear at the top of the `.fetchmailrc` file, followed by one or more server options, each of which designate a different email server that Fetchmail should check. User options follow server options for each user account checking that email server. Like server options, multiple user options may be specified for use with a particular server as well as to check multiple email accounts on the same server.

Server options are called into service in the `.fetchmailrc` file by the use of a special option verb, `poll` or `skip`, that precedes any of the server information. The `poll` action tells Fetchmail to use this server option when it is run, which checks for email using the specified user options. Any server options after a `skip` action, however, are not checked unless this server's hostname is specified when Fetchmail is invoked. The `skip` option is useful when testing configurations in the `.fetchmailrc` file because it only checks skipped servers when specifically invoked, and does not affect any currently working configurations.

The following is a sample example of a `.fetchmailrc` file:

```
set postmaster "user1"
set bouncemail

poll pop.domain.com proto pop3
    user 'user1' there with password 'secret' is user1 here

poll mail.domain2.com
    user 'user5' there with password 'secret2' is user1 here
    user 'user7' there with password 'secret3' is user1 here
```

In this example, the global options specify that the user is sent email as a last resort (`postmaster` option) and all email errors are sent to the postmaster instead of the sender (`bouncemail` option). The `set` action tells Fetchmail that this line contains a global option. Then, two email servers are specified, one set to check using **POP3**, the other for trying various protocols to find one that works. Two users are checked using the second server option, but all email found for any user is sent to `user1`'s mail spool. This allows multiple mailboxes to be checked on multiple servers, while appearing in a single MUA inbox. Each user's specific information begins with the `user` action.

**Note**

Users are not required to place their password in the `.fetchmailrc` file. Omitting the `with password '<password>'` section causes Fetchmail to ask for a password when it is launched.

Fetchmail has numerous global, server, and local options. Many of these options are rarely used or only apply to very specific situations. The `fetchmail` man page explains each option in detail, but the most common ones are listed in the following three sections.

10.3.3.2. Global Options

Each global option should be placed on a single line after a `set` action.

- ▶ **daemon <seconds>** — Specifies daemon-mode, where Fetchmail stays in the background. Replace `<seconds>` with the number of seconds Fetchmail is to wait before polling the server.
- ▶ **postmaster** — Specifies a local user to send mail to in case of delivery problems.
- ▶ **syslog** — Specifies the log file for errors and status messages. By default, this is `/var/log/maillog`.

10.3.3.3. Server Options

Server options must be placed on their own line in `.fetchmailrc` after a `poll` or `skip` action.

- ▶ **auth <auth-type>** — Replace `<auth-type>` with the type of authentication to be used. By default, `password` authentication is used, but some protocols support other types of authentication, including `kerberos_v5`, `kerberos_v4`, and `ssh`. If the `any` authentication type is used, Fetchmail first tries methods that do not require a password, then methods that mask the password, and finally attempts to send the password unencrypted to authenticate to the server.
- ▶ **interval <number>** — Polls the specified server every `<number>` of times that it checks for email on all configured servers. This option is generally used for email servers where the user rarely receives messages.
- ▶ **port <port-number>** — Replace `<port-number>` with the port number. This value overrides the default port number for the specified protocol.
- ▶ **proto <protocol>** — Replace `<protocol>` with the protocol, such as `pop3` or `imap`, to use when checking for messages on the server.
- ▶ **timeout <seconds>** — Replace `<seconds>` with the number of seconds of server inactivity after which Fetchmail gives up on a connection attempt. If this value is not set, a default of **300** seconds is used.

10.3.3.4. User Options

User options may be placed on their own lines beneath a server option or on the same line as the server option. In either case, the defined options must follow the `user` option (defined below).

- ▶ **fetchall** — Orders Fetchmail to download all messages in the queue, including messages that have already been viewed. By default, Fetchmail only pulls down new messages.

- ▶ **fetchlimit <number>** — Replace <number> with the number of messages to be retrieved before stopping.
- ▶ **flush** — Deletes all previously viewed messages in the queue before retrieving new messages.
- ▶ **limit <max-number-bytes>** — Replace <max-number-bytes> with the maximum size in bytes that messages are allowed to be when retrieved by Fetchmail. This option is useful with slow network links, when a large message takes too long to download.
- ▶ **password '<password>'** — Replace <password> with the user's password.
- ▶ **preconnect "<command>"** — Replace <command> with a command to be executed before retrieving messages for the user.
- ▶ **postconnect "<command>"** — Replace <command> with a command to be executed after retrieving messages for the user.
- ▶ **ssl** — Activates SSL encryption.
- ▶ **user "<username>"** — Replace <username> with the username used by Fetchmail to retrieve messages. *This option must precede all other user options.*

10.3.3.5. Fetchmail Command Options

Most Fetchmail options used on the command line when executing the **fetchmail** command mirror the **.fetchmailrc** configuration options. In this way, Fetchmail may be used with or without a configuration file. These options are not used on the command line by most users because it is easier to leave them in the **.fetchmailrc** file.

There may be times when it is desirable to run the **fetchmail** command with other options for a particular purpose. It is possible to issue command options to temporarily override a **.fetchmailrc** setting that is causing an error, as any options specified at the command line override configuration file options.

10.3.3.6. Informational or Debugging Options

Certain options used after the **fetchmail** command can supply important information.

- ▶ **--configdump** — Displays every possible option based on information from **.fetchmailrc** and Fetchmail defaults. No email is retrieved for any users when using this option.
- ▶ **-s** — Executes Fetchmail in silent mode, preventing any messages, other than errors, from appearing after the **fetchmail** command.
- ▶ **-v** — Executes Fetchmail in verbose mode, displaying every communication between Fetchmail and remote email servers.
- ▶ **-V** — Displays detailed version information, lists its global options, and shows settings to be used with each user, including the email protocol and authentication method. No email is retrieved for any users when using this option.

10.3.3.7. Special Options

These options are occasionally useful for overriding defaults often found in the **.fetchmailrc** file.

- ▶ **-a** — Fetchmail downloads all messages from the remote email server, whether new or previously viewed. By default, Fetchmail only downloads new messages.

- ▶ **-k** — Fetchmail leaves the messages on the remote email server after downloading them. This option overrides the default behavior of deleting messages after downloading them.
- ▶ **-l <max-number-bytes>** — Fetchmail does not download any messages over a particular size and leaves them on the remote email server.
- ▶ **--quit** — Quits the Fetchmail daemon process.

More commands and `.fetchmailrc` options can be found in the **fetchmail** man page.

10.3.4. Mail Transport Agent (MTA) Configuration

A *Mail Transport Agent* (MTA) is essential for sending email. A *Mail User Agent* (MUA) such as **Evolution** or **Mutt**, is used to read and compose email. When a user sends an email from an MUA, the message is handed off to the MTA, which sends the message through a series of MTAs until it reaches its destination.

Even if a user does not plan to send email from the system, some automated tasks or system programs might use the **mail** command to send email containing log messages to the **root** user of the local system.

Red Hat Enterprise Linux 7 provides two MTAs: Postfix and Sendmail. If both are installed, Postfix is the default MTA. Note that Sendmail is considered deprecated in Red Hat Enterprise Linux 7.

10.4. Mail Delivery Agents

Red Hat Enterprise Linux includes two primary MDAs, Procmail and **mail**. Both of the applications are considered LDAs and both move email from the MTA's spool file into the user's mailbox. However, Procmail provides a robust filtering system.

This section details only Procmail. For information on the **mail** command, consult its man page (**man mail**).

Procmail delivers and filters email as it is placed in the mail spool file of the localhost. It is powerful, gentle on system resources, and widely used. Procmail can play a critical role in delivering email to be read by email client applications.

Procmail can be invoked in several different ways. Whenever an MTA places an email into the mail spool file, Procmail is launched. Procmail then filters and files the email for the MUA and quits. Alternatively, the MUA can be configured to execute Procmail any time a message is received so that messages are moved into their correct mailboxes. By default, the presence of `/etc/procmailrc` or of a `~/.procmailrc` file (also called an `rc` file) in the user's home directory invokes Procmail whenever an MTA receives a new message.

By default, no system-wide `rc` files exist in the `/etc/` directory and no `.procmailrc` files exist in any user's home directory. Therefore, to use Procmail, each user must construct a `.procmailrc` file with specific environment variables and rules.

Whether Procmail acts upon an email message depends upon whether the message matches a specified set of conditions or *recipes* in the `rc` file. If a message matches a recipe, then the email is placed in a specified file, is deleted, or is otherwise processed.

When Procmail starts, it reads the email message and separates the body from the header information. Next, Procmail looks for a `/etc/procmailrc` file and `rc` files in the `/etc/procmailrcs` directory for default, system-wide, Procmail environmental variables and recipes. Procmail then searches for a `.procmailrc` file in the user's home directory. Many users also create additional `rc` files for Procmail that are referred to within the `.procmailrc` file in their home directory.

10.4.1. Procmail Configuration

The Procmail configuration file contains important environmental variables. These variables specify things such as which messages to sort and what to do with the messages that do not match any recipes.

These environmental variables usually appear at the beginning of the `~/.procmailrc` file in the following format:

```
<env-variable>=<value>
```

In this example, `<env-variable>` is the name of the variable and `<value>` defines the variable.

There are many environment variables not used by most Procmail users and many of the more important environment variables are already defined by a default value. Most of the time, the following variables are used:

- ▶ **DEFAULT** — Sets the default mailbox where messages that do not match any recipes are placed.

The default **DEFAULT** value is the same as **\$ORGMAIL**.

- ▶ **INCLUDERC** — Specifies additional `rc` files containing more recipes for messages to be checked against. This breaks up the Procmail recipe lists into individual files that fulfill different roles, such as blocking spam and managing email lists, that can then be turned off or on by using comment characters in the user's `~/.procmailrc` file.

For example, lines in a user's `.procmailrc` file may look like this:

```
MAILDIR=$HOME/Msgs INCLUDERC=$MAILDIR/lists.rc INCLUDERC=$MAILDIR/spam.rc
```

To turn off Procmail filtering of email lists but leaving spam control in place, comment out the first **INCLUDERC** line with a hash sign (#).

- ▶ **LOCKSLEEP** — Sets the amount of time, in seconds, between attempts by Procmail to use a particular lockfile. The default is **8** seconds.
- ▶ **LOCKTIMEOUT** — Sets the amount of time, in seconds, that must pass after a lockfile was last modified before Procmail assumes that the lockfile is old and can be deleted. The default is **1024** seconds.
- ▶ **LOGFILE** — The file to which any Procmail information or error messages are written.
- ▶ **MAILDIR** — Sets the current working directory for Procmail. If set, all other Procmail paths are relative to this directory.
- ▶ **ORGMAIL** — Specifies the original mailbox, or another place to put the messages if they cannot be placed in the default or recipe-required location.

By default, a value of `/var/spool/mail/$LOGNAME` is used.

- ▶ **SUSPEND** — Sets the amount of time, in seconds, that Procmail pauses if a necessary resource, such as swap space, is not available.
- ▶ **SWITCHRC** — Allows a user to specify an external file containing additional Procmail recipes, much like the **INCLUDERC** option, except that recipe checking is actually stopped on the referring configuration file and only the recipes on the **SWITCHRC**-specified file are used.
- ▶ **VERBOSE** — Causes Procmail to log more information. This option is useful for debugging.

Other important environmental variables are pulled from the shell, such as **LOGNAME**, the login name; **HOME**, the location of the home directory; and **SHELL**, the default shell.

A comprehensive explanation of all environments variables, and their default values, is available in the **procmailrc** man page.

10.4.2. Procmal Recipes

New users often find the construction of recipes the most difficult part of learning to use Procmal. This difficulty is often attributed to recipes matching messages by using *regular expressions* which are used to specify qualifications for string matching. However, regular expressions are not very difficult to construct and even less difficult to understand when read. Additionally, the consistency of the way Procmal recipes are written, regardless of regular expressions, makes it easy to learn by example. To see example Procmal recipes, see [Section 10.4.2.5, “Recipe Examples”](#).

Procmal recipes take the following form:

```
:0<flags>: <lockfile-name> * <special-condition-character>
    <condition-1> * <special-condition-character>
    <condition-2> * <special-condition-character>
    <condition-N>
    <special-action-character>
    <action-to-perform>
```

The first two characters in a Procmal recipe are a colon and a zero. Various flags can be placed after the zero to control how Procmal processes the recipe. A colon after the **<flags>** section specifies that a lockfile is created for this message. If a lockfile is created, the name can be specified by replacing **<lockfile-name>**.

A recipe can contain several conditions to match against the message. If it has no conditions, every message matches the recipe. Regular expressions are placed in some conditions to facilitate message matching. If multiple conditions are used, they must all match for the action to be performed. Conditions are checked based on the flags set in the recipe's first line. Optional special characters placed after the asterisk character (*) can further control the condition.

The **<action-to-perform>** argument specifies the action taken when the message matches one of the conditions. There can only be one action per recipe. In many cases, the name of a mailbox is used here to direct matching messages into that file, effectively sorting the email. Special action characters may also be used before the action is specified. See [Section 10.4.2.4, “Special Conditions and Actions”](#) for more information.

10.4.2.1. Delivering vs. Non-Delivering Recipes

The action used if the recipe matches a particular message determines whether it is considered a *delivering* or *non-delivering* recipe. A delivering recipe contains an action that writes the message to a file, sends the message to another program, or forwards the message to another email address. A non-delivering recipe covers any other actions, such as a *nesting block*. A nesting block is a set of actions, contained in braces { }, that are performed on messages which match the recipe's conditions. Nesting blocks can be nested inside one another, providing greater control for identifying and performing actions on messages.

When messages match a delivering recipe, Procmal performs the specified action and stops comparing the message against any other recipes. Messages that match non-delivering recipes continue to be compared against other recipes.

10.4.2.2. Flags

Flags are essential to determine how or if a recipe's conditions are compared to a message. The following flags are commonly used:

- ▶ **A** — Specifies that this recipe is only used if the previous recipe without an **A** or **a** flag also matched this message.
- ▶ **a** — Specifies that this recipe is only used if the previous recipe with an **A** or **a** flag also matched this message *and* was successfully completed.
- ▶ **B** — Parses the body of the message and looks for matching conditions.
- ▶ **b** — Uses the body in any resulting action, such as writing the message to a file or forwarding it. This is the default behavior.
- ▶ **c** — Generates a carbon copy of the email. This is useful with delivering recipes, since the required action can be performed on the message and a copy of the message can continue being processed in the **rc** files.
- ▶ **D** — Makes the **egrep** comparison case-sensitive. By default, the comparison process is not case-sensitive.
- ▶ **E** — While similar to the **A** flag, the conditions in the recipe are only compared to the message if the immediately preceding recipe without an **E** flag did not match. This is comparable to an **else** action.
- ▶ **e** — The recipe is compared to the message only if the action specified in the immediately preceding recipe fails.
- ▶ **f** — Uses the pipe as a filter.
- ▶ **H** — Parses the header of the message and looks for matching conditions. This is the default behavior.
- ▶ **h** — Uses the header in a resulting action. This is the default behavior.
- ▶ **w** — Tells Procmail to wait for the specified filter or program to finish, and reports whether or not it was successful before considering the message filtered.
- ▶ **W** — Is identical to **w** except that "Program failure" messages are suppressed.

For a detailed list of additional flags, see the **procmailrc** man page.

10.4.2.3. Specifying a Local Lockfile

Lockfiles are very useful with Procmail to ensure that more than one process does not try to alter a message simultaneously. Specify a local lockfile by placing a colon (:) after any flags on a recipe's first line. This creates a local lockfile based on the destination file name plus whatever has been set in the **LOCKEXT** global environment variable.

Alternatively, specify the name of the local lockfile to be used with this recipe after the colon.

10.4.2.4. Special Conditions and Actions

Special characters used before Procmail recipe conditions and actions change the way they are interpreted.

The following characters may be used after the asterisk character (*) at the beginning of a recipe's condition line:

- ▶ **!** — In the condition line, this character inverts the condition, causing a match to occur only if the

condition does not match the message.

- ▶ < — Checks if the message is under a specified number of bytes.
- ▶ > — Checks if the message is over a specified number of bytes.

The following characters are used to perform special actions:

- ▶ ! — In the action line, this character tells Procmail to forward the message to the specified email addresses.
- ▶ \$ — Refers to a variable set earlier in the **rc** file. This is often used to set a common mailbox that is referred to by various recipes.
- ▶ | — Starts a specified program to process the message.
- ▶ { and } — Constructs a nesting block, used to contain additional recipes to apply to matching messages.

If no special character is used at the beginning of the action line, Procmail assumes that the action line is specifying the mailbox in which to write the message.

10.4.2.5. Recipe Examples

Procmail is an extremely flexible program, but as a result of this flexibility, composing Procmail recipes from scratch can be difficult for new users.

The best way to develop the skills to build Procmail recipe conditions stems from a strong understanding of regular expressions combined with looking at many examples built by others. A thorough explanation of regular expressions is beyond the scope of this section. The structure of Procmail recipes and useful sample Procmail recipes can be found at various places on the Internet. The proper use and adaptation of regular expressions can be derived by viewing these recipe examples. In addition, introductory information about basic regular expression rules can be found in the **grep** man page.

The following simple examples demonstrate the basic structure of Procmail recipes and can provide the foundation for more intricate constructions.

A basic recipe may not even contain conditions, as is illustrated in the following example:

```
:0: new-mail.spool
```

The first line specifies that a local lockfile is to be created but does not specify a name, so Procmail uses the destination file name and appends the value specified in the **LOCKEXT** environment variable. No condition is specified, so every message matches this recipe and is placed in the single spool file called **new-mail.spool**, located within the directory specified by the **MAILDIR** environment variable. An MUA can then view messages in this file.

A basic recipe, such as this, can be placed at the end of all **rc** files to direct messages to a default location.

The following example matched messages from a specific email address and throws them away.

```
:0 * ^From: spammer@domain.com /dev/null
```

With this example, any messages sent by **spammer@domain.com** are sent to the **/dev/null** device, deleting them.



Warning

Be certain that rules are working as intended before sending messages to `/dev/null` for permanent deletion. If a recipe inadvertently catches unintended messages, and those messages disappear, it becomes difficult to troubleshoot the rule.

A better solution is to point the recipe's action to a special mailbox, which can be checked from time to time to look for false positives. Once satisfied that no messages are accidentally being matched, delete the mailbox and direct the action to send the messages to `/dev/null`.

The following recipe grabs email sent from a particular mailing list and places it in a specified folder.

```
:0: * ^{From|Cc|To}.* tux-lug tuxlug
```

Any messages sent from the `tux-lug@domain.com` mailing list are placed in the `tuxlug` mailbox automatically for the MUA. Note that the condition in this example matches the message if it has the mailing list's email address on the `From`, `Cc`, or `To` lines.

Consult the many Procmail online resources available in [Section 10.6, “Additional Resources”](#) for more detailed and powerful recipes.

10.4.2.6. Spam Filters

Because it is called by Sendmail, Postfix, and Fetchmail upon receiving new emails, Procmail can be used as a powerful tool for combating spam.

This is particularly true when Procmail is used in conjunction with SpamAssassin. When used together, these two applications can quickly identify spam emails, and sort or destroy them.

SpamAssassin uses header analysis, text analysis, blacklists, a spam-tracking database, and self-learning Bayesian spam analysis to quickly and accurately identify and tag spam.



Note

In order to use **SpamAssassin**, first ensure the `spamassassin` package is installed on your system by running, as `root`:

```
~]# yum install spamassassin
```

For more information on installing packages with Yum, see [Section 5.2.4, “Installing Packages”](#).

The easiest way for a local user to use SpamAssassin is to place the following line near the top of the `~/.procmailrc` file:

```
INCLUDERC=/etc/mail/spamassassin/spamassassin-default.rc
```

The `/etc/mail/spamassassin/spamassassin-default.rc` contains a simple Procmail rule that activates SpamAssassin for all incoming email. If an email is determined to be spam, it is tagged in the header as such and the title is prepended with the following pattern:

```
* * * * * SPAM* * * *
```

The message body of the email is also prepended with a running tally of what elements caused it to be diagnosed as spam.

To file email tagged as spam, a rule similar to the following can be used:

```
:0 Hw * ^X-Spam-Status: Yes spam
```

This rule files all email tagged in the header as spam into a mailbox called **spam**.

Since SpamAssassin is a Perl script, it may be necessary on busy servers to use the binary SpamAssassin daemon (**spamd**) and the client application (**spamc**). Configuring SpamAssassin this way, however, requires **root** access to the host.

To start the **spamd** daemon, type the following command:

```
~]# systemctl start spamassassin
```

To start the SpamAssassin daemon when the system is booted, use an initscript utility, such as the **Services Configuration Tool (system-config-services)**, to turn on the **spamassassin** service. See [Chapter 6, Managing Services with systemd](#) for more information about starting and stopping services.

To configure Procmail to use the SpamAssassin client application instead of the Perl script, place the following line near the top of the `~/.procmailrc` file. For a system-wide configuration, place it in `/etc/procmailrc`:

```
INCLUDERC=/etc/mail/spamassassin/spamassassin-spamc.rc
```

10.5. Mail User Agents

Red Hat Enterprise Linux offers a variety of email programs, both, graphical email client programs, such as **Evolution**, and text-based email programs such as **Mutt**.

The remainder of this section focuses on securing communication between a client and a server.

10.5.1. Securing Communication

Popular MUAs included with Red Hat Enterprise Linux, such as **Evolution** and **Mutt** offer SSL-encrypted email sessions.

Like any other service that flows over a network unencrypted, important email information, such as usernames, passwords, and entire messages, may be intercepted and viewed by users on the network. Additionally, since the standard **POP** and **IMAP** protocols pass authentication information unencrypted, it is possible for an attacker to gain access to user accounts by collecting usernames and passwords as they are passed over the network.

10.5.1.1. Secure Email Clients

Most Linux MUAs designed to check email on remote servers support SSL encryption. To use SSL when retrieving email, it must be enabled on both the email client and the server.

SSL is easy to enable on the client-side, often done with the click of a button in the MUA's configuration window or via an option in the MUA's configuration file. Secure **IMAP** and **POP** have known port numbers (**993** and **995**, respectively) that the MUA uses to authenticate and download messages.

10.5.1.2. Securing Email Client Communications

Offering SSL encryption to **IMAP** and **POP** users on the email server is a simple matter.

First, create an SSL certificate. This can be done in two ways: by applying to a *Certificate Authority (CA)* for an SSL certificate or by creating a self-signed certificate.



Warning

Self-signed certificates should be used for testing purposes only. Any server used in a production environment should use an SSL certificate granted by a CA.

To create a self-signed SSL certificate for **IMAP** or **POP**, change to the `/etc/pki/dovecot/` directory, edit the certificate parameters in the `/etc/pki/dovecot/dovecot-openssl.cnf` configuration file as you prefer, and type the following commands, as **root**:

```
dovecot]# rm -f certs/dovecot.pem private/dovecot.pem
dovecot]# /usr/libexec/dovecot/mkcert.sh
```

Once finished, make sure you have the following configurations in your `/etc/dovecot/conf.d/10-ssl.conf` file:

```
ssl_cert = </etc/pki/dovecot/certs/dovecot.pem
ssl_key = </etc/pki/dovecot/private/dovecot.pem
```

Execute the following command to restart the **dovecot** daemon:

```
~]# systemctl restart dovecot
```

Alternatively, the **stunnel** command can be used as an SSL encryption wrapper around the standard, non-secure connections to **IMAP** or **POP** services.

The **stunnel** utility uses external OpenSSL libraries included with Red Hat Enterprise Linux to provide strong cryptography and to protect the network connections. It is recommended to apply to a CA to obtain an SSL certificate, but it is also possible to create a self-signed certificate.



Note

In order to use **stunnel**, first ensure the *stunnel* package is installed on your system by running, as **root**:

```
~]# yum install stunnel
```

For more information on installing packages with Yum, see [Section 5.2.4, “Installing Packages”](#).

To create a self-signed SSL certificate, change to the `/etc/pki/tls/certs/` directory, and type the following command:

```
certs]# make stunnel.pem
```

Answer all of the questions to complete the process.

Once the certificate is generated, create an **stunnel** configuration file, for example **/etc/stunnel/mail.conf**, with the following content:

```
cert = /etc/pki/tls/certs/stunnel.pem

[pop3s]
accept = 995
connect = 110

[imaps]
accept = 993
connect = 143
```

Once you start **stunnel** with the created configuration file using the **stunnel /etc/stunnel/mail.conf** command, it will be possible to use an **IMAP** or a **POP** email client and connect to the email server using SSL encryption.

For more information on **stunnel**, see the **stunnel** man page or the documents in the **/usr/share/doc/stunnel-<version-number>** / directory, where <version-number> is the version number of **stunnel**.

10.6. Additional Resources

The following is a list of additional documentation about email applications.

10.6.1. Installed Documentation

- ▶ Information on configuring Sendmail is included with the **sendmail** and **sendmail-cf** packages.
 - **/usr/share/sendmail-cf/README** — Contains information on the **m4** macro processor, file locations for Sendmail, supported mailers, how to access enhanced features, and more.
- ▶ In addition, the **sendmail** and **aliases** man pages contain helpful information covering various Sendmail options and the proper configuration of the Sendmail **/etc/mail/aliases** file.
- ▶ **/usr/share/doc/postfix-<version-number>** — Contains a large amount of information about ways to configure Postfix. Replace <version-number> with the version number of Postfix.
- ▶ **/usr/share/doc/fetchmail-<version-number>** — Contains a full list of Fetchmail features in the **FEATURES** file and an introductory **FAQ** document. Replace <version-number> with the version number of Fetchmail.
- ▶ **/usr/share/doc/procmail-<version-number>** — Contains a **README** file that provides an overview of Procmail, a **FEATURES** file that explores every program feature, and an **FAQ** file with answers to many common configuration questions. Replace <version-number> with the version number of Procmail.

When learning how Procmail works and creating new recipes, the following Procmail man pages are invaluable:

- **procmail** — Provides an overview of how Procmail works and the steps involved with filtering email.
- **procmailrc** — Explains the **rc** file format used to construct recipes.

- **procmailex** — Gives a number of useful, real-world examples of Procmail recipes.
- **procmailsc** — Explains the weighted scoring technique used by Procmail to match a particular recipe to a message.
- **/usr/share/doc/spamassassin-<version-number>/** — Contains a large amount of information pertaining to SpamAssassin. Replace <version-number> with the version number of the **spamassassin** package.

10.6.2. Useful Websites

- ▶ <http://www.sendmail.org/> — Offers a thorough technical breakdown of Sendmail features, documentation and configuration examples.
- ▶ <http://www.sendmail.com/> — Contains news, interviews and articles concerning Sendmail, including an expanded view of the many options available.
- ▶ <http://www.postfix.org/> — The Postfix project home page contains a wealth of information about Postfix. The mailing list is a particularly good place to look for information.
- ▶ <http://fetchmail.berlios.de/> — The home page for Fetchmail, featuring an online manual, and a thorough FAQ.
- ▶ <http://www.procmail.org/> — The home page for Procmail with links to assorted mailing lists dedicated to Procmail as well as various FAQ documents.
- ▶ <http://www.uwasa.fi/~ts/info/proctips.html> — Contains dozens of tips that make using Procmail much easier. Includes instructions on how to test **.procmailrc** files and use Procmail scoring to decide if a particular action should be taken.
- ▶ <http://www.spamassassin.org/> — The official site of the SpamAssassin project.

10.6.3. Related Books

- ▶ *Sendmail Filters: A Guide for Fighting Spam* by Bryan Costales and Marcia Flynt; Addison-Wesley — A good Sendmail guide that can help you customize your mail filters.
- ▶ *Sendmail* by Bryan Costales with Eric Allman et al.; O'Reilly & Associates — A good Sendmail reference written with the assistance of the original creator of Delivermail and Sendmail.
- ▶ *Removing the Spam: Email Processing and Filtering* by Geoff Mulligan; Addison-Wesley Publishing Company — A volume that looks at various methods used by email administrators using established tools, such as Sendmail and Procmail, to manage spam problems.
- ▶ *Internet Email Protocols: A Developer's Guide* by Kevin Johnson; Addison-Wesley Publishing Company — Provides a very thorough review of major email protocols and the security they provide.
- ▶ *Managing IMAP* by Dianna Mullet and Kevin Mullet; O'Reilly & Associates — Details the steps required to configure an IMAP server.

Chapter 11. Directory Servers

11.1. OpenLDAP

LDAP (Lightweight Directory Access Protocol) is a set of open protocols used to access centrally stored information over a network. It is based on the **X.500** standard for directory sharing, but is less complex and resource-intensive. For this reason, LDAP is sometimes referred to as “X.500 Lite”.

Like X.500, LDAP organizes information in a hierarchical manner using directories. These directories can store a variety of information such as names, addresses, or phone numbers, and can even be used in a manner similar to the *Network Information Service* (NIS), enabling anyone to access their account from any machine on the LDAP enabled network.

LDAP is commonly used for centrally managed users and groups, user authentication, or system configuration. It can also serve as a virtual phone directory, allowing users to easily access contact information for other users. Additionally, it can refer a user to other LDAP servers throughout the world, and thus provide an ad-hoc global repository of information. However, it is most frequently used within individual organizations such as universities, government departments, and private companies.

This section covers the installation and configuration of **OpenLDAP 2.4**, an open source implementation of the LDAPv2 and LDAPv3 protocols.

11.1.1. Introduction to LDAP

Using a client/server architecture, LDAP provides reliable means to create a central information directory accessible from the network. When a client attempts to modify information within this directory, the server verifies the user has permission to make the change, and then adds or updates the entry as requested. To ensure the communication is secure, the *Secure Sockets Layer* (SSL) or *Transport Layer Security* (TLS) cryptographic protocols can be used to prevent an attacker from intercepting the transmission.



Using Mozilla NSS

The OpenLDAP suite in Red Hat Enterprise Linux 7 no longer uses OpenSSL. Instead, it uses the Mozilla implementation of *Network Security Services* (NSS). OpenLDAP continues to work with existing certificates, keys, and other TLS configuration. For more information on how to configure it to use Mozilla certificate and key database, see [How do I use TLS/SSL with Mozilla NSS](#).

The LDAP server supports several database systems, which gives administrators the flexibility to choose the best suited solution for the type of information they are planning to serve. Because of a well-defined client *Application Programming Interface* (API), the number of applications able to communicate with an LDAP server is numerous, and increasing in both quantity and quality.

11.1.1.1. LDAP Terminology

The following is a list of LDAP-specific terms that are used within this chapter:

entry

A single unit within an LDAP directory. Each entry is identified by its unique *Distinguished Name* (DN).

attribute

Information directly associated with an entry. For example, if an organization is represented as an LDAP entry, attributes associated with this organization might include an address, a fax number, etc. Similarly, people can be represented as entries with common attributes such as personal telephone number or email address.

An attribute can either have a single value, or an unordered space-separated list of values. While certain attributes are optional, others are required. Required attributes are specified using the **objectClass** definition, and can be found in schema files located in the `/etc/openldap/slapd.d/cn=config/cn=schema/` directory.

The assertion of an attribute and its corresponding value is also referred to as a *Relative Distinguished Name* (RDN). Unlike distinguished names that are unique globally, a relative distinguished name is only unique per entry.

LDIF

The *LDAP Data Interchange Format* (LDIF) is a plain text representation of an LDAP entry. It takes the following form:

```
[id] dn: distinguished_name
attribute_type: attribute_value...
attribute_type: attribute_value...
...
...
```

The optional *id* is a number determined by the application that is used to edit the entry. Each entry can contain as many *attribute_type* and *attribute_value* pairs as needed, as long as they are all defined in a corresponding schema file. A blank line indicates the end of an entry.

11.1.1.2. OpenLDAP Features

OpenLDAP suite provides a number of important features:

- ▶ *LDAPv3 Support* — Many of the changes in the protocol since LDAP version 2 are designed to make LDAP more secure. Among other improvements, this includes the support for Simple Authentication and Security Layer (SASL), Transport Layer Security (TLS), and Secure Sockets Layer (SSL) protocols.
- ▶ *LDAP Over IPC* — The use of inter-process communication (IPC) enhances security by eliminating the need to communicate over a network.
- ▶ *IPv6 Support* — OpenLDAP is compliant with Internet Protocol version 6 (IPv6), the next generation of the Internet Protocol.
- ▶ *LDIFv1 Support* — OpenLDAP is fully compliant with LDIF version 1.
- ▶ *Updated C API* — The current C API improves the way programmers can connect to and use LDAP directory servers.
- ▶ *Enhanced Standalone LDAP Server* — This includes an updated access control system, thread pooling, better tools, and much more.

11.1.1.3. OpenLDAP Server Setup

The typical steps to set up an LDAP server on Red Hat Enterprise Linux are as follows:

1. Install the OpenLDAP suite. See [Section 11.1.2, “Installing the OpenLDAP Suite”](#) for more information on required packages.
2. Customize the configuration as described in [Section 11.1.3, “Configuring an OpenLDAP Server”](#).

3. Start the **slapd** service as described in [Section 11.1.4, “Running an OpenLDAP Server”](#).
4. Use the **ldapadd** utility to add entries to the LDAP directory.
5. Use the **ldapsearch** utility to verify that the **slapd** service is accessing the information correctly.

11.1.2. Installing the OpenLDAP Suite

The suite of OpenLDAP libraries and tools is provided by the following packages:

Table 11.1. List of OpenLDAP packages

Package	Description
<i>openldap</i>	A package containing the libraries necessary to run the OpenLDAP server and client applications.
<i>openldap-clients</i>	A package containing the command line utilities for viewing and modifying directories on an LDAP server.
<i>openldap-servers</i>	A package containing both the services and utilities to configure and run an LDAP server. This includes the <i>Standalone LDAP Daemon</i> , slapd .
<i>compat-openldap</i>	A package containing the OpenLDAP compatibility libraries.

Additionally, the following packages are commonly used along with the LDAP server:

Table 11.2. List of commonly installed additional LDAP packages

Package	Description
<i>nss-pam-ldapd</i>	A package containing ns1cd , a local LDAP name service that allows a user to perform local LDAP queries.

To install these packages, use the **yum** command in the following form:

```
yum install package...
```

For example, to perform the basic LDAP server installation, type the following at a shell prompt:

```
~]# yum install openldap openldap-clients openldap-servers
```

Note that you must have superuser privileges (that is, you must be logged in as **root**) to run this command. For more information on how to install new packages in Red Hat Enterprise Linux, see [Section 5.2.4, “Installing Packages”](#).

11.1.2.1. Overview of OpenLDAP Server Utilities

To perform administrative tasks, the *openldap-servers* package installs the following utilities along with the **slapd** service:

Table 11.3. List of OpenLDAP server utilities

Command	Description
slapac1	Allows you to check the access to a list of attributes.
slapadd	Allows you to add entries from an LDIF file to an LDAP directory.

Command	Description
slapauth	Allows you to check a list of IDs for authentication and authorization permissions.
slapcat	Allows you to pull entries from an LDAP directory in the default format and save them in an LDIF file.
slapdn	Allows you to check a list of Distinguished Names (DNs) based on available schema syntax.
slapindex	Allows you to re-index the slapd directory based on the current content. Run this utility whenever you change indexing options in the configuration file.
slappasswd	Allows you to create an encrypted user password to be used with the ldapmodify utility, or in the slapd configuration file.
slapschema	Allows you to check the compliance of a database with the corresponding schema.
slaptest	Allows you to check the LDAP server configuration.

For a detailed description of these utilities and their usage, see the corresponding manual pages as referred to in [Section 11.1.6.1, “Installed Documentation”](#).



Make sure the files have correct owner

Although only **root** can run **slapadd**, the **slapd** service runs as the **ldap** user. Because of this, the directory server is unable to modify any files created by **slapadd**. To correct this issue, after running the **slapd** utility, type the following at a shell prompt:

```
~]# chown -R ldap:ldap /var/lib/ldap
```



Stop slapd before using these utilities

To preserve the data integrity, stop the **slapd** service before using **slapadd**, **slapcat**, or **slapindex**. You can do so by typing the following at a shell prompt:

```
~]# systemctl stop slapd.service
```

For more information on how to start, stop, restart, and check the current status of the **slapd** service, see [Section 11.1.4, “Running an OpenLDAP Server”](#).

11.1.2.2. Overview of OpenLDAP Client Utilities

The *openldap-clients* package installs the following utilities which can be used to add, modify, and delete entries in an LDAP directory:

Table 11.4. List of OpenLDAP client utilities

Command	Description
ldapadd	Allows you to add entries to an LDAP directory, either from a file, or from standard input. It is a symbolic link to ldapmodify -a .

Command	Description
ldapcompare	Allows you to compare given attribute with an LDAP directory entry.
ldapdelete	Allows you to delete entries from an LDAP directory.
ldapexop	Allows you to perform extended LDAP operations.
ldapmodify	Allows you to modify entries in an LDAP directory, either from a file, or from standard input.
ldapmodrdn	Allows you to modify the RDN value of an LDAP directory entry.
ldappasswd	Allows you to set or change the password for an LDAP user.
ldapsearch	Allows you to search LDAP directory entries.
ldapurl	Allows you to compose or decompose LDAP URLs.
ldapwhoami	Allows you to perform a whoami operation on an LDAP server.

With the exception of **ldapsearch**, each of these utilities is more easily used by referencing a file containing the changes to be made rather than typing a command for each entry to be changed within an LDAP directory. The format of such a file is outlined in the man page for each utility.

11.1.2.3. Overview of Common LDAP Client Applications

Although there are various graphical LDAP clients capable of creating and modifying directories on the server, none of them is included in Red Hat Enterprise Linux. Popular applications that can access directories in a read-only mode include **Mozilla Thunderbird**, **Evolution**, or **Ekiga**.

11.1.3. Configuring an OpenLDAP Server

By default, the OpenLDAP configuration is stored in the **/etc/openldap/** directory. The following table highlights the most important directories and files within this directory:

Table 11.5. List of OpenLDAP configuration files and directories

Path	Description
/etc/openldap/ldap.conf	The configuration file for client applications that use the OpenLDAP libraries. This includes ldapadd , ldapsearch , Evolution , etc.
/etc/openldap/slapd.d/	The directory containing the slapd configuration.

Note that OpenLDAP no longer reads its configuration from the **/etc/openldap/slapd.conf** file. Instead, it uses a configuration database located in the **/etc/openldap/slapd.d/** directory. If you have an existing **slapd.conf** file from a previous installation, you can convert it to the new format by running the following command:

```
~]# slaptest -f /etc/openldap/slapd.conf -F /etc/openldap/slapd.d/
```

The **slapd** configuration consists of LDIF entries organized in a hierarchical directory structure, and the recommended way to edit these entries is to use the server utilities described in [Section 11.1.2.1, “Overview of OpenLDAP Server Utilities”](#).



Do not edit LDIF files directly

An error in an LDIF file can render the **slapd** service unable to start. Because of this, it is strongly advised that you avoid editing the LDIF files within the **/etc/openldap/slapd.d/** directly.

11.1.3.1. Changing the Global Configuration

Global configuration options for the LDAP server are stored in the `/etc/openldap/slapd.d/cn=config.ldif` file. The following directives are commonly used:

olcAllows

The **olcAllows** directive allows you to specify which features to enable. It takes the following form:

```
olcAllows: feature...
```

It accepts a space-separated list of features as described in [Table 11.6, “Available olcAllows options”](#). The default option is **bind_v2**.

Table 11.6. Available olcAllows options

Option	Description
bind_v2	Enables the acceptance of LDAP version 2 bind requests.
bind_anon_cred	Enables an anonymous bind when the Distinguished Name (DN) is empty.
bind_anon_dn	Enables an anonymous bind when the Distinguished Name (DN) is <i>not</i> empty.
update_anon	Enables processing of anonymous update operations.
proxy_authz_anon	Enables processing of anonymous proxy authorization control.

Example 11.1. Using the **olcAllows** directive

```
olcAllows: bind_v2 update_anon
```

olcConnMaxPending

The **olcConnMaxPending** directive allows you to specify the maximum number of pending requests for an anonymous session. It takes the following form:

```
olcConnMaxPending: number
```

The default option is **100**.

Example 11.2. Using the **olcConnMaxPending** directive

```
olcConnMaxPending: 100
```

olcConnMaxPendingAuth

The **olcConnMaxPendingAuth** directive allows you to specify the maximum number of pending requests for an authenticated session. It takes the following form:

```
olcConnMaxPendingAuth: number
```

The default option is **1000**.

Example 11.3. Using the olcConnMaxPendingAuth directive

```
olcConnMaxPendingAuth: 1000
```

olcDisallows

The **olcDisallows** directive allows you to specify which features to disable. It takes the following form:

```
olcDisallows: feature...
```

It accepts a space-separated list of features as described in [Table 11.7, “Available olcDisallows options”](#). No features are disabled by default.

Table 11.7. Available olcDisallows options

Option	Description
bind_anon	Disables the acceptance of anonymous bind requests.
bind_simple	Disables the simple bind authentication mechanism.
tls_2_anon	Disables the enforcing of an anonymous session when the STARTTLS command is received.
tls_authc	Disallows the STARTTLS command when authenticated.

Example 11.4. Using the olcDisallows directive

```
olcDisallows: bind_anon
```

olcIdleTimeout

The **olcIdleTimeout** directive allows you to specify how many seconds to wait before closing an idle connection. It takes the following form:

```
olcIdleTimeout: number
```

This option is disabled by default (that is, set to **0**).

Example 11.5. Using the olcIdleTimeout directive

```
olcIdleTimeout: 180
```

olcLogFile

The **olcLogFile** directive allows you to specify a file in which to write log messages. It takes the following form:

```
olcLogFile: file_name
```

The log messages are written to standard error by default.

Example 11.6. Using the olcLogFile directive

```
olcLogFile: /var/log/slapd.log
```

olcReferral

The **olcReferral** option allows you to specify a URL of a server to process the request in case the server is not able to handle it. It takes the following form:

```
olcReferral: URL
```

This option is disabled by default.

Example 11.7. Using the olcReferral directive

```
olcReferral: ldap://root.openldap.org
```

olcWriteTimeout

The **olcWriteTimeout** option allows you to specify how many seconds to wait before closing a connection with an outstanding write request. It takes the following form:

```
olcWriteTimeout
```

This option is disabled by default (that is, set to **0**).

Example 11.8. Using the olcWriteTimeout directive

```
olcWriteTimeout: 180
```

11.1.3.2. Changing the Database-Specific Configuration

By default, the OpenLDAP server uses Berkeley DB (BDB) as a database back end. The configuration for this database is stored in the **/etc/openldap/slapd.d/cn=config/olcDatabase={1}bdb.ldif** file. The following directives are commonly used in a database-specific configuration:

olcReadOnly

The **olcReadOnly** directive allows you to use the database in a read-only mode. It takes the following form:

```
olcReadOnly: boolean
```

It accepts either **TRUE** (enable the read-only mode), or **FALSE** (enable modifications of the database). The default option is **FALSE**.

Example 11.9. Using the olcReadOnly directive

```
olcReadOnly: TRUE
```

olcRootDN

The **olcRootDN** directive allows you to specify the user that is unrestricted by access controls or administrative limit parameters set for operations on the LDAP directory. It takes the following form:

```
olcRootDN: distinguished_name
```

It accepts a *Distinguished Name* (DN). The default option is **cn=Manager, dn=my-domain, dc=com**.

Example 11.10. Using the olcRootDN directive

```
olcRootDN: cn=root, dn=example, dn=com
```

olcRootPW

The **olcRootPW** directive allows you to set a password for the user that is specified using the **olcRootDN** directive. It takes the following form:

```
olcRootPW: password
```

It accepts either a plain text string, or a hash. To generate a hash, type the following at a shell prompt:

```
~]$ slappasswd
New password:
Re-enter new password:
{SSHA}WczWsyPEnMchFf1GRTweq2q7XJcvmSxD
```

Example 11.11. Using the olcRootPW directive

```
olcRootPW: {SSHA}WczWsyPEnMchFf1GRTweq2q7XJcvmSxD
```

olcSuffix

The **olcSuffix** directive allows you to specify the domain for which to provide information. It takes the following form:

```
olcSuffix: domain_name
```

It accepts a *fully qualified domain name* (FQDN). The default option is **dc=my-domain, dc=com**.

Example 11.12. Using the olcSuffix directive

```
olcSuffix: dc=example, dc=com
```

11.1.3.3. Extending Schema

Since OpenLDAP 2.3, the **/etc/openldap/slapd.d/** directory also contains LDAP definitions that were previously located in **/etc/openldap/schema/**. It is possible to extend the schema used by OpenLDAP to support additional attribute types and object classes using the default schema files as a guide. However, this task is beyond the scope of this chapter. For more information on this topic, see

<http://www.openldap.org/doc/admin/schema.html>

11.1.4. Running an OpenLDAP Server

This section describes how to start, stop, restart, and check the current status of the **Standalone LDAP Daemon**. For more information on how to manage system services in general, see [Chapter 6, Managing Services with systemd](#).

11.1.4.1. Starting the Service

To start the **slapd** service in the current session, type the following at a shell prompt as **root**:

```
~]# systemctl start slapd.service
```

To configure the service to start automatically at the boot time, use the following command as **root**:

```
~]# systemctl enable slapd.service
ln -s '/usr/lib/systemd/system/slapd.service' '/etc/systemd/system/multi-user.target.wants/slapd.service'
```

11.1.4.2. Stopping the Service

To stop the running **slapd** service in the current session, type the following at a shell prompt as **root**:

```
~]# systemctl stop slapd.service
```

To prevent the service from starting automatically at the boot time, type as **root**:

```
~]# systemctl disable slapd.service
rm '/etc/systemd/system/multi-user.target.wants/slapd.service'
```

11.1.4.3. Restarting the Service

To restart the running **slapd** service, type the following at a shell prompt:

```
~]# systemctl restart slapd.service
```

This stops the service and immediately starts it again. Use this command to reload the configuration.

11.1.4.4. Verifying the Service Status

To verify that the **slapd** service is running, type the following at a shell prompt:

```
~]# systemctl is-active slapd.service
active
```

11.1.5. Configuring a System to Authenticate Using OpenLDAP

In order to configure a system to authenticate using OpenLDAP, make sure that the appropriate packages are installed on both LDAP server and client machines. For information on how to set up the server, follow the instructions in [Section 11.1.2, “Installing the OpenLDAP Suite”](#) and [Section 11.1.3, “Configuring an OpenLDAP Server”](#). On a client, type the following at a shell prompt:

```
~]# yum install openldap openldap-clients nss-pam-ldapd
```

For detailed instructions on how to configure applications to use LDAP for authentication, see the *Red Hat Enterprise Linux 7 Authentication Guide*.

11.1.5.1. Migrating Old Authentication Information to LDAP Format

The *migrationtools* package provides a set of shell and Perl scripts to help you migrate authentication information into an LDAP format. To install this package, type the following at a shell prompt:

```
~]# yum install migrationtools
```

This will install the scripts to the `/usr/share/migrationtools/` directory. Once installed, edit the `/usr/share/migrationtools/migrate_common.ph` file and change the following lines to reflect the correct domain, for example:

```
# Default DNS domain
$DEFAULT_MAIL_DOMAIN = "example.com";

# Default base
$DEFAULT_BASE = "dc=example,dc=com";
```

Alternatively, you can specify the environment variables directly on the command line. For example, to run the `migrate_all_online.sh` script with the default base set to `dc=example,dc=com`, type:

```
~]# export DEFAULT_BASE="dc=example,dc=com" \
/usr/share/migrationtools/migrate_all_online.sh
```

To decide which script to run in order to migrate the user database, see [Table 11.8, “Commonly used LDAP migration scripts”](#).

Table 11.8. Commonly used LDAP migration scripts

Existing Name Service	Is LDAP Running?	Script to Use
/etc flat files	yes	<code>migrate_all_online.sh</code>
/etc flat files	no	<code>migrate_all_offline.sh</code>
NetInfo	yes	<code>migrate_all_netinfo_online.sh</code>
NetInfo	no	<code>migrate_all_netinfo_offline.sh</code>
NIS (YP)	yes	<code>migrate_all_nis_online.sh</code>
NIS (YP)	no	<code>migrate_all_nis_offline.sh</code>

For more information on how to use these scripts, see the `README` and the `migration-tools.txt` files in the `/usr/share/doc/migrationtools-version/` directory.

11.1.6. Additional Resources

The following resources offer additional information on the Lightweight Directory Access Protocol. Before configuring LDAP on your system, it is highly recommended that you review these resources, especially the *OpenLDAP Software Administrator's Guide*.

11.1.6.1. Installed Documentation

The following documentation is installed with the `openldap-servers` package:

/usr/share/doc/openldap-servers-version/guide.html

A copy of the *OpenLDAP Software Administrator's Guide*.

/usr/share/doc/openldap-servers-version/README.schema

A README file containing the description of installed schema files.

Additionally, there is also a number of manual pages that are installed with the *openldap*, *openldap-servers*, and *openldap-clients* packages:

Client Applications

- ▶ **man ldapadd** — Describes how to add entries to an LDAP directory.
- ▶ **man ldapdelete** — Describes how to delete entries within an LDAP directory.
- ▶ **man ldapmodify** — Describes how to modify entries within an LDAP directory.
- ▶ **man ldapsearch** — Describes how to search for entries within an LDAP directory.
- ▶ **man ldappasswd** — Describes how to set or change the password of an LDAP user.
- ▶ **man ldapcompare** — Describes how to use the **ldapcompare** tool.
- ▶ **man ldapwhoami** — Describes how to use the **ldapwhoami** tool.
- ▶ **man ldapmodrdn** — Describes how to modify the RDNs of entries.

Server Applications

- ▶ **man slapd** — Describes command line options for the LDAP server.

Administrative Applications

- ▶ **man slapadd** — Describes command line options used to add entries to a **slapd** database.
- ▶ **man slapcat** — Describes command line options used to generate an LDIF file from a **slapd** database.
- ▶ **man slapindex** — Describes command line options used to regenerate an index based upon the contents of a **slapd** database.
- ▶ **man slappasswd** — Describes command line options used to generate user passwords for LDAP directories.

Configuration Files

- ▶ **man ldap.conf** — Describes the format and options available within the configuration file for LDAP clients.
- ▶ **man slapd-config** — Describes the format and options available within the configuration directory.

11.1.6.2. Useful Websites

<http://www.openldap.org/doc/admin24/>

The current version of the *OpenLDAP Software Administrator's Guide*.

<http://www.kingsmountain.com/ldapRoadmap.shtml>

Jeff Hodges' *LDAP Roadmap & FAQ* containing links to several useful resources and emerging news concerning the LDAP protocol.

<http://wwwldapman.org/articles/>

A collection of articles that offer a good introduction to LDAP, including methods to design a directory tree and customizing directory structures.

<http://www.padl.com/>

A website of developers of several useful LDAP tools.

11.1.6.3. Related Books

***OpenLDAP by Example* by John Terpstra and Benjamin Coles; Prentice Hall.**

A collection of practical exercises in the OpenLDAP deployment.

***Implementing LDAP* by Mark Wilcox; Wrox Press, Inc.**

A book covering LDAP from both the system administrator's and software developer's perspective.

***Understanding and Deploying LDAP Directory Services* by Tim Howes et al.; Macmillan Technical Publishing.**

A book covering LDAP design principles, as well as its deployment in a production environment.

Chapter 12. File and Print Servers

12.1. Samba

Samba is an open source implementation of the *server message block* (**SMB**) protocol. Modern versions of this protocol are also known as the *common Internet file system* (**CIFS**) protocol. It allows the networking of Microsoft Windows®, Linux, UNIX, and other operating systems together, enabling access to Windows-based file and printer shares. Samba's use of **SMB** allows it to appear as a Windows server to Windows clients.



Note

In order to use **Samba**, first ensure the *samba* package is installed on your system by running, as **root**:

```
~]# yum install samba
```

For more information on installing packages with Yum, see [Section 5.2.4, “Installing Packages”](#).

12.1.1. Introduction to Samba

Red Hat Enterprise Linux 7 includes Samba version 4 .1:

12.1.1.1. Samba Features

Samba is a powerful and versatile server application.

What Samba can do:

- ▶ Serve directory trees and printers to Linux, UNIX, and Windows clients
- ▶ Assist in network browsing (with or without NetBIOS)
- ▶ Authenticate Windows domain logins
- ▶ Provide *Windows Internet Name Service* (**WINS**) name server resolution
- ▶ Act as a Windows NT®-style *Primary Domain Controller* (PDC)
- ▶ Act as a *Backup Domain Controller* (BDC) for a Samba-based PDC
- ▶ Act as an Active Directory domain member server
- ▶ Join a Windows NT/2000/2003/2008 PDC/Windows Server 2012

What Samba cannot do:

- ▶ Act as a BDC for a Windows PDC (and vice versa)
- ▶ Act as an Active Directory domain controller

12.1.2. Samba Daemons and Related Services

The following is a brief introduction to the individual Samba daemons and services.

12.1.2.1. Samba Daemons

Samba is comprised of three daemons (**smbd**, **nmbd**, and **winbindd**). Three services (**smb**, **nmb**, and **winbind**) control how the daemons are started, stopped, and other service-related features. These services act as different init scripts. Each daemon is listed in detail below, as well as which specific service has control over it.

smbd

The **smbd** server daemon provides file sharing and printing services to Windows clients. In addition, it is responsible for user authentication, resource locking, and data sharing through the **SMB** protocol. The default ports on which the server listens for **SMB** traffic are **TCP** ports **139** and **445**.

The **smbd** daemon is controlled by the **smb** service.

nmbd

The **nmbd** server daemon understands and replies to NetBIOS name service requests such as those produced by SMB/CIFS in Windows-based systems. These systems include Windows 95/98/ME, Windows NT, Windows 2000, Windows XP, and LanManager clients. It also participates in the browsing protocols that make up the Windows **Network Neighborhood** view. The default port that the server listens to for **NMB** traffic is **UDP** port **137**.

The **nmbd** daemon is controlled by the **nmb** service.

winbindd

The **winbind** service resolves user and group information received from a server running Windows NT, 2000, 2003, Windows Server 2008, or Windows Server 2012. This makes Windows user and group information understandable by UNIX platforms. This is achieved by using Microsoft RPC calls, *Pluggable Authentication Modules* (PAM), and the *Name Service Switch* (NSS). This allows Windows NT domain users to appear and operate as UNIX users on a UNIX machine. Though bundled with the Samba distribution, the **winbind** service is controlled separately from the **smb** service.

The **winbindd** daemon is controlled by the **winbind** service and does not require the **smb** service to be started in order to operate. **winbindd** is also used when Samba is an Active Directory member, and may also be used on a Samba domain controller (to implement nested groups and interdomain trust). Because **winbind** is a client-side service used to connect to Windows NT-based servers, further discussion of **winbind** is beyond the scope of this chapter.



Note

See [Section 12.1.7, “Samba Distribution Programs”](#) for a list of utilities included in the Samba distribution.

12.1.3. Connecting to a Samba Share

You can use **Nautilus** to view available Samba shares on your network. To view a list of Samba workgroups and domains on your network, select **Places** → **Network** from the GNOME panel, and select your desired network. You can also type **smb:** in the **File** → **Open Location** bar of **Nautilus** to view the workgroups/domains.

As shown in [Figure 12.1, “SMB Workgroups in Nautilus”](#), an icon appears for each available **SMB** workgroup or domain on the network.

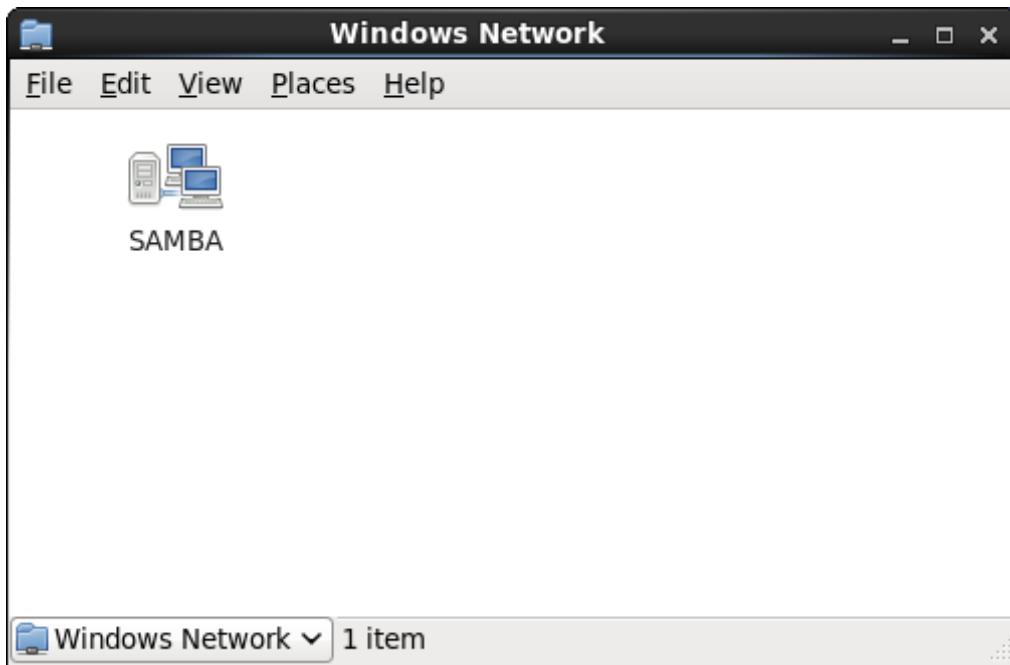


Figure 12.1. SMB Workgroups in Nautilus

Double-click one of the workgroup/domain icons to view a list of computers within the workgroup/domain.

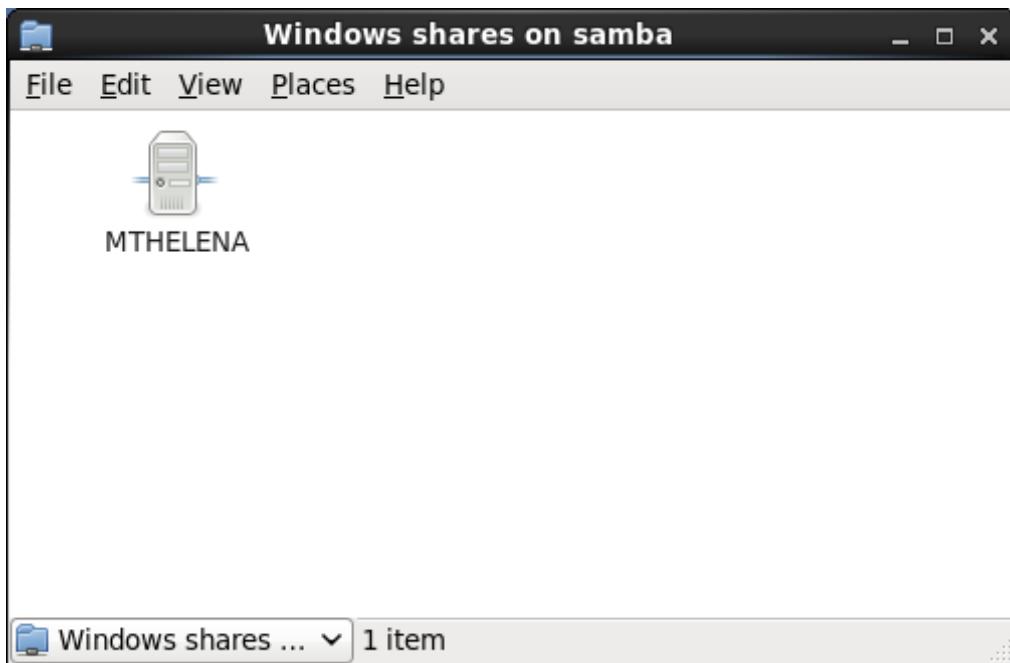


Figure 12.2. SMB Machines in Nautilus

As displayed in [Figure 12.2, “SMB Machines in Nautilus”](#), an icon exists for each machine within the workgroup. Double-click on an icon to view the Samba shares on the machine. If a user name and password combination is required, you are prompted for them.

Alternately, you can also specify the Samba server and sharename in the **Location:** bar for **Nautilus** using the following syntax (replace <servername> and <sharename> with the appropriate values):

```
smb://<servername>/<sharename>
```

12.1.3.1. Command Line

To connect to a Samba share from a shell prompt, type the following command:

```
~]$ smbclient //<hostname>/<sharename> -U <username>
```

Replace *<hostname>* with the hostname or **IP** address of the Samba server you want to connect to, *<sharename>* with the name of the shared directory you want to browse, and *<username>* with the Samba username for the system. Enter the correct password or press **Enter** if no password is required for the user.

If you see the **smb:>** prompt, you have successfully logged in. Once you are logged in, type **help** for a list of commands. If you wish to browse the contents of your home directory, replace *sharename* with your username. If the **-U** switch is not used, the username of the current user is passed to the Samba server.

To exit **smbclient**, type **exit** at the **smb:>** prompt.

12.1.3.2. Mounting the Share

Sometimes it is useful to mount a Samba share to a directory so that the files in the directory can be treated as if they are part of the local file system.

To mount a Samba share to a directory, create a directory to mount it to (if it does not already exist), and execute the following command as **root**:

```
~]# mount -t cifs //<servername>/<sharename> /mnt/point/ -o  
username=<username>,password=<password>
```

This command mounts *<sharename>* from *<servername>* in the local directory */mnt/point/*.

Note

The **mount.cifs** utility is a separate RPM (independent from Samba). In order to use **mount.cifs**, first ensure the *cifs-utils* package is installed on your system by running, as **root**:

```
~]# yum install cifs-utils
```

For more information on installing packages with Yum, see [Section 5.2.4, “Installing Packages”](#).

Note that the *cifs-utils* package also contains the **cifs.upcall** binary called by the kernel in order to perform kerberized CIFS mounts. For more information on **cifs.upcall**, see **man cifs.upcall**.

For more information about mounting a samba share, see **man mount.cifs**.



Warning

Some CIFS servers require plain text passwords for authentication. Support for plain text password authentication can be enabled using the following command:

```
~]# echo 0x37 > /proc/fs/cifs/SecurityFlags
```

WARNING: This operation can expose passwords by removing password encryption.

12.1.4. Configuring a Samba Server

The default configuration file (**/etc/samba/smb.conf**) allows users to view their home directories as a Samba share. It also shares all printers configured for the system as Samba shared printers. You can attach a printer to the system and print to it from the Windows machines on your network.

12.1.4.1. Graphical Configuration

To configure Samba using a graphical interface, use one of the available Samba graphical user interfaces. A list of available GUIs can be found at <http://www.samba.org/samba/GUI/>.

12.1.4.2. Command Line Configuration

Samba uses **/etc/samba/smb.conf** as its configuration file. If you change this configuration file, the changes do not take effect until you restart the Samba daemon with the following command, as **root**:

```
~]# systemctl restart smb.service
```

To specify the Windows workgroup and a brief description of the Samba server, edit the following lines in your **/etc/samba/smb.conf** file:

```
workgroup = WORKGROUPNAME
server string = BRIEF COMMENT ABOUT SERVER
```

Replace **WORKGROUPNAME** with the name of the Windows workgroup to which this machine should belong. The **BRIEF COMMENT ABOUT SERVER** is optional and is used as the Windows comment about the Samba system.

To create a Samba share directory on your Linux system, add the following section to your **/etc/samba/smb.conf** file (after modifying it to reflect your needs and your system):

```
[sharename]
comment = Insert a comment here
path = /home/share/
valid users = tfox carole
public = no
writable = yes
printable = no
create mask = 0765
```

The above example allows the users **tfox** and **carole** to read and write to the directory **/home/share**, on the Samba server, from a Samba client.

12.1.4.3. Encrypted Passwords

Encrypted passwords are enabled by default because it is more secure to use them. To create a user with an encrypted password, use the command **smbpasswd -a <username>**.

12.1.5. Starting and Stopping Samba

To start a Samba server, type the following command in a shell prompt, as **root**:

```
~]# systemctl start smb.service
```



Important

To set up a domain member server, you must first join the domain or Active Directory using the **net join** command before starting the **smb** service. Also it is recommended to run **winbind** before **smbd**.

To stop the server, type the following command in a shell prompt, as **root**:

```
~]# systemctl stop smb.service
```

The **restart** option is a quick way of stopping and then starting Samba. This is the most reliable way to make configuration changes take effect after editing the configuration file for Samba. Note that the restart option starts the daemon even if it was not running originally.

To restart the server, type the following command in a shell prompt, as **root**:

```
~]# systemctl restart smb.service
```

The **condrestart** (*conditional restart*) option only starts **smb** on the condition that it is currently running. This option is useful for scripts, because it does not start the daemon if it is not running.



Note

When the **/etc/samba/smb.conf** file is changed, Samba automatically reloads it after a few minutes. Issuing a manual **restart** or **reload** is just as effective.

To conditionally restart the server, type the following command, as **root**:

```
~]# systemctl try-restart smb.service
```

A manual reload of the **/etc/samba/smb.conf** file can be useful in case of a failed automatic reload by the **smb** service. To ensure that the Samba server configuration file is reloaded without restarting the service, type the following command, as **root**:

```
~]# systemctl reload smb.service
```

By default, the **smb** service does *not* start automatically at boot time. To configure Samba to start at boot time, type the following at a shell prompt as **root**:

```
~]# systemctl enable smb.service
```

See [Chapter 6, Managing Services with `systemd`](#) for more information regarding these tools.

12.1.6. Samba Network Browsing

Network browsing enables Windows and Samba servers to appear in the Windows **Network Neighborhood**. Inside the **Network Neighborhood**, icons are represented as servers and if opened, the server's shares and printers that are available are displayed.

Network browsing capabilities require NetBIOS over **TCP/IP**. NetBIOS-based networking uses broadcast (**UDP**) messaging to accomplish browse list management. Without NetBIOS and WINS as the primary method for **TCP/IP** hostname resolution, other methods such as static files (**/etc/hosts**) or **DNS**, must be used.

A domain master browser collates the browse lists from local master browsers on all subnets so that browsing can occur between workgroups and subnets. Also, the domain master browser should preferably be the local master browser for its own subnet.

12.1.6.1. Domain Browsing

By default, a Windows server PDC for a domain is also the domain master browser for that domain. A Samba server must *not* be set up as a domain master server in this type of situation.

For subnets that do not include the Windows server PDC, a Samba server can be implemented as a local master browser. Configuring the **/etc/samba/smb.conf** file for a local master browser (or no browsing at all) in a domain controller environment is the same as workgroup configuration (see [Section 12.1.4, "Configuring a Samba Server"](#)).

12.1.6.2. WINS (Windows Internet Name Server)

Either a Samba server or a Windows NT server can function as a WINS server. When a WINS server is used with NetBIOS enabled, UDP unicasts can be routed which allows name resolution across networks. Without a WINS server, the UDP broadcast is limited to the local subnet and therefore cannot be routed to other subnets, workgroups, or domains. If WINS replication is necessary, do not use Samba as your primary WINS server, as Samba does not currently support WINS replication.

In a mixed NT/2000/2003/2008 server and Samba environment, it is recommended that you use the Microsoft WINS capabilities. In a Samba-only environment, it is recommended that you use *only one* Samba server for WINS.

The following is an example of the **/etc/samba/smb.conf** file in which the Samba server is serving as a WINS server:

```
[global]
wins support = Yes
```

Note

All servers (including Samba) should connect to a WINS server to resolve NetBIOS names. Without WINS, browsing only occurs on the local subnet. Furthermore, even if a domain-wide list is somehow obtained, hosts cannot be resolved for the client without WINS.

12.1.7. Samba Distribution Programs

net

net <protocol> <function> <misc_options> <target_options>

The **net** utility is similar to the **net** utility used for Windows and MS-DOS. The first argument is used to specify the protocol to use when executing a command. The **<protocol>** option can be **ads**, **rap**, or **rpc** for specifying the type of server connection. Active Directory uses **ads**, Win9x/NT3 uses **rap**, and Windows NT4/2000/2003/2008 uses **rpc**. If the protocol is omitted, **net** automatically tries to determine it.

The following example displays a list of the available shares for a host named **wakko**:

```
~]$ net -l share -S wakko
Password:
Enumerating shared resources (exports) on remote server:
Share name      Type      Description
-----          -----
data            Disk      Wakko data share
tmp             Disk      Wakko tmp share
IPC$           IPC       IPC Service (Samba Server)
ADMIN$         IPC       IPC Service (Samba Server)
```

The following example displays a list of Samba users for a host named **wakko**:

```
~]$ net -l user -S wakko
root password:
User name          Comment
-----
andriusb          Documentation
joe               Marketing
lisa              Sales
```

nmblookup**nmblookup <options> <netbios_name>**

The **nmblookup** program resolves NetBIOS names into **IP** addresses. The program broadcasts its query on the local subnet until the target machine replies.

The following example displays the **IP** address of the NetBIOS name **trek**:

```
~]$ nmblookup trek
querying trek on 10.1.59.255
10.1.56.45 trek<00>
```

pdbedit**pdbedit <options>**

The **pdbedit** program manages accounts located in the SAM database. All back ends are supported including **smbpasswd**, LDAP, and the **tdb** database library.

The following are examples of adding, deleting, and listing users:

```
~]$ pdbedit -a kristin
new password:
retype new password:
Unix username:      kristin
NT username:
Account Flags:      [U      ]
```

```
User SID: S-1-5-21-1210235352-3804200048-1474496110-2012
Primary Group SID: S-1-5-21-1210235352-3804200048-1474496110-2077
Full Name: Home Directory: \\wakko\kristin
HomeDir Drive:
Logon Script:
Profile Path: \\wakko\kristin\profile
Domain: WAKKO
Account desc:
Workstations: Munged
dial:
Logon time: 0
Logoff time: Mon, 18 Jan 2038 22:14:07 GMT
Kickoff time: Mon, 18 Jan 2038 22:14:07 GMT
Password last set: Thu, 29 Jan 2004 08:29:28
GMT Password can change: Thu, 29 Jan 2004 08:29:28 GMT
Password must change: Mon, 18 Jan 2038 22:14:07 GMT
~]$ pdbedit -v -L kristin
Unix username: kristin
NT username:
Account Flags: [U      ]
User SID: S-1-5-21-1210235352-3804200048-1474496110-2012
Primary Group SID: S-1-5-21-1210235352-3804200048-1474496110-2077
Full Name:
Home Directory: \\wakko\kristin
HomeDir Drive:
Logon Script:
Profile Path: \\wakko\kristin\profile
Domain: WAKKO
Account desc:
Workstations: Munged
dial:
Logon time: 0
Logoff time: Mon, 18 Jan 2038 22:14:07 GMT
Kickoff time: Mon, 18 Jan 2038 22:14:07 GMT
Password last set: Thu, 29 Jan 2004 08:29:28 GMT
Password can change: Thu, 29 Jan 2004 08:29:28 GMT
Password must change: Mon, 18 Jan 2038 22:14:07 GMT
~]$ pdbedit -L
andriusb:505:
joe:503:
lisa:504:
kristin:506:
~]$ pdbedit -x joe
~]$ pdbedit -L
andriusb:505: lisa:504: kristin:506:
```

rpcclient

rpcclient <server> <options>

The **rpcclient** program issues administrative commands using Microsoft RPCs, which provide access to the Windows administration graphical user interfaces (GUIs) for systems management. This is most often used by advanced users that understand the full complexity of Microsoft RPCs.

smbcacls

smbcacls </>server/share> <filename> <options>

The **smbcacls** program modifies Windows ACLs on files and directories shared by a Samba server or a Windows server.

smbclient**smbclient** </>server/share> <password> <options>The **smbclient** program is a versatile UNIX client which provides functionality similar to **ftp**.**smbcontrol****smbcontrol** -i <options>**smbcontrol** <options> <destination> <messagetype> <parameters>The **smbcontrol** program sends control messages to running **smbd**, **nmbd**, or **winbinddd** daemons. Executing **smbcontrol** -i runs commands interactively until a blank line or a 'q' is entered.**smbpasswd****smbpasswd** <options> <username> <password>The **smbpasswd** program manages encrypted passwords. This program can be run by a superuser to change any user's password and also by an ordinary user to change their own Samba password.**smbspool****smbspool** <job> <user> <title> <copies> <options> <filename>The **smbspool** program is a CUPS-compatible printing interface to Samba. Although designed for use with CUPS printers, **smbspool** can work with non-CUPS printers as well.**smbstatus****smbstatus** <options>The **smbstatus** program displays the status of current connections to a Samba server.**smbtar****smbtar** <options>The **smbtar** program performs backup and restores of Windows-based share files and directories to a local tape archive. Though similar to the **tar** command, the two are not compatible.**testparm****testparm** <options> <filename> <hostname IP_address>The **testparm** program checks the syntax of the **/etc/samba/smb.conf** file. If your **smb.conf** file is in the default location (**/etc/samba/smb.conf**) you do not need to specify the location. Specifying the host name and **IP** address to the **testparm** program verifies that the **hosts.allow** and **host.deny** files are configured correctly. The **testparm** program also displays a summary of your **smb.conf** file and the server's role (stand-alone, domain, etc.) after testing. This is convenient when debugging as it excludes comments and concisely presents information for experienced administrators to read.

For example:

```

~]$ testparm
Load smb config files from /etc/samba/smb.conf
Processing section "[homes]"
Processing section "[printers]"
Processing section "[tmp]"
Processing section "[html]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions
<enter>
# Global parameters
[global]
  workgroup = MYGROUP
  server string = Samba Server
  security = SHARE
  log file = /var/log/samba/%m.log
  max log size = 50
  socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192
  dns proxy = No
[homes]
  comment = Home Directories
  read only = No
  browseable = No
[printers]
  comment = All Printers
  path = /var/spool/samba
  printable = Yes
  browseable = No
[tmp]
  comment = Wakko tmp
  path = /tmp
  guest only = Yes
[html]
  comment = Wakko www
  path = /var/www/html
  force user = andriusb
  force group = users
  read only = No
  guest only = Yes

```

wbinfo

wbinfo <options>

The **wbinfo** program displays information from the **winbindd** daemon. The **winbindd** daemon must be running for **wbinfo** to work.

12.1.8. Additional Resources

The following sections give you the means to explore Samba in greater detail.

12.1.8.1. Installed Documentation



Note

In order to use the **Samba** documentation, first ensure the *samba* package is installed on your system by running, as **root**:

```
~]# yum install samba
```

For more information on installing packages with Yum, see [Section 5.2.4, “Installing Packages”](#).

/usr/share/doc/samba-<version-number>/ — All additional files included with the Samba distribution. This includes all helper scripts, sample configuration files, and documentation.

See the following man pages for detailed information specific **Samba** features:

- **smb.conf**
- **samba**
- **smbd**
- **nmbd**
- **winbind**

12.1.8.2. Useful Websites

- ▶ <http://www.samba.org/> — Homepage for the Samba distribution and all official documentation created by the Samba development team. Many resources are available in HTML and PDF formats, while others are only available for purchase. Although many of these links are not Red Hat Enterprise Linux specific, some concepts may apply.
- ▶ <http://samba.org/samba/archives.html> — Active email lists for the Samba community. Enabling digest mode is recommended due to high levels of list activity.
- ▶ Samba newsgroups — Samba threaded newsgroups, such as www.gmane.org, that use the **NNTP** protocol are also available. This an alternative to receiving mailing list emails.

12.2. FTP

The *File Transfer Protocol (FTP)* is one of the oldest and most commonly used protocols found on the Internet today. Its purpose is to reliably transfer files between computer hosts on a network without requiring the user to log directly in to the remote host or to have knowledge of how to use the remote system. It allows users to access files on remote systems using a standard set of simple commands.

This section outlines the basics of the **FTP** protocol and introduces **vsftpd**, the primary **FTP** server shipped with Red Hat Enterprise Linux.

12.2.1. The File Transfer Protocol

FTP uses a client-server architecture to transfer files using the **TCP** network protocol. Because **FTP** is a rather old protocol, it uses unencrypted user name and password authentication. For this reason, it is considered an insecure protocol and should not be used unless absolutely necessary. However, because **FTP** is so prevalent on the Internet, it is often required for sharing files to the public. System administrators, therefore, should be aware of **FTP**'s unique characteristics.

This section describes how to configure **vsftpd** to establish connections secured by **SSL** and how to secure an **FTP** server with the help of **SELinux**. A good substitute for **FTP** is **sftp** from the **OpenSSH** suite of tools. For information about configuring **OpenSSH** and about the **SSH** protocol in general, refer to [Chapter 7, OpenSSH](#).

Unlike most protocols used on the Internet, **FTP** requires multiple network ports to work properly. When an **FTP** client application initiates a connection to an **FTP** server, it opens port **21** on the server — known as the *command port*. This port is used to issue all commands to the server. Any data requested from the server is returned to the client via a *data port*. The port number for data connections, and the way in which data connections are initialized, vary depending upon whether the client requests the data in *active* or *passive* mode.

The following defines these modes:

active mode

Active mode is the original method used by the **FTP** protocol for transferring data to the client application. When an active-mode data transfer is initiated by the **FTP** client, the server opens a connection from port **20** on the server to the **IP** address and a random, unprivileged port (greater than **1024**) specified by the client. This arrangement means that the client machine must be allowed to accept connections over any port above **1024**. With the growth of insecure networks, such as the Internet, the use of firewalls for protecting client machines is now prevalent. Because these client-side firewalls often deny incoming connections from active-mode **FTP** servers, passive mode was devised.

passive mode

Passive mode, like active mode, is initiated by the **FTP** client application. When requesting data from the server, the **FTP** client indicates it wants to access the data in passive mode and the server provides the **IP** address and a random, unprivileged port (greater than **1024**) on the server. The client then connects to that port on the server to download the requested information.

While passive mode does resolve issues for client-side firewall interference with data connections, it can complicate administration of the server-side firewall. You can reduce the number of open ports on a server by limiting the range of unprivileged ports on the **FTP** server. This also simplifies the process of configuring firewall rules for the server.

12.2.2. The vsftpd Server

The Very Secure FTP Daemon (vsftpd) is designed from the ground up to be fast, stable, and, most importantly, secure. **vsftpd** is the only stand-alone **FTP** server distributed with Red Hat Enterprise Linux, due to its ability to handle large numbers of connections efficiently and securely.

The security model used by **vsftpd** has three primary aspects:

- ▶ *Strong separation of privileged and non-privileged processes* — Separate processes handle different tasks, and each of these processes runs with the minimal privileges required for the task.

- ▶ Tasks requiring elevated privileges are handled by processes with the minimal privilege necessary — By taking advantage of compatibilities found in the **libcap** library, tasks that usually require full root privileges can be executed more safely from a less privileged process.
- ▶ Most processes run in a **chroot** jail — Whenever possible, processes are change-rooted to the directory being shared; this directory is then considered a **chroot** jail. For example, if the **/var/ftp** directory is the primary shared directory, **vsftpd** reassigns **/var/ftp/** to the new root directory, known as **/**. This disallows any potential malicious hacker activities for any directories not contained in the new root directory.

Use of these security practices has the following effect on how **vsftpd** deals with requests:

- ▶ The parent process runs with the least privileges required — The parent process dynamically calculates the level of privileges it requires to minimize the level of risk. Child processes handle direct interaction with the **FTP** clients and run with as close to no privileges as possible.
- ▶ All operations requiring elevated privileges are handled by a small parent process — Much like the **Apache HTTP Server**, **vsftpd** launches unprivileged child processes to handle incoming connections. This allows the privileged, parent process to be as small as possible and handle relatively few tasks.
- ▶ All requests from unprivileged child processes are distrusted by the parent process — Communication with child processes is received over a socket, and the validity of any information from child processes is checked before being acted on.
- ▶ Most interactions with **FTP** clients are handled by unprivileged child processes in a **chroot** jail — Because these child processes are unprivileged and only have access to the directory being shared, any crashed processes only allow the attacker access to the shared files.

12.2.2.1. Starting and Stopping **vsftpd**

To start the **vsftpd** service in the current session, type the following at a shell prompt as **root**:

```
~]# systemctl start vsftpd.service
```

To stop the service in the current session, type as **root**:

```
~]# systemctl stop vsftpd.service
```

To restart the **vsftpd** service, run the following command as **root**:

```
~]# systemctl restart vsftpd.service
```

This command stops and immediately starts the **vsftpd** service, which is the most efficient way to make configuration changes take effect after editing the configuration file for this **FTP** server. Alternatively, you can use the following command to restart the **vsftpd** service only if it is already running:

```
~]# systemctl try-restart vsftpd.service
```

By default, the **vsftpd** service does *not* start automatically at boot time. To configure the **vsftpd** service to start at boot time, type the following at a shell prompt as **root**:

```
~]# systemctl enable vsftpd.service
ln -s '/usr/lib/systemd/system/vsftpd.service' '/etc/systemd/system/multi-
user.target.wants/vsftpd.service'
```

For more information on how to manage system services in Red Hat Enterprise Linux 7, see [Chapter 6, Managing Services with `systemd`](#).

12.2.2.2. Starting Multiple Copies of `vsftpd`

Sometimes, one computer is used to serve multiple **FTP** domains. This is a technique called *multihoming*. One way to multihome using **vsftpd** is by running multiple copies of the daemon, each with its own configuration file.

To do this, first assign all relevant **IP** addresses to network devices or alias network devices on the system. For more information about configuring network devices, device aliases, and additional information about network configuration scripts, see the [Red Hat Enterprise Linux 7 Networking Guide](#).

Next, the **DNS** server for the **FTP** domains must be configured to reference the correct machine. For information about **BIND**, the **DNS** protocol implementation used in Red Hat Enterprise Linux, and its configuration files, see the [Red Hat Enterprise Linux 7 Networking Guide](#).

For **vsftpd** to answer requests on different **IP** addresses, multiple copies of the daemon must be running. To facilitate launching multiple instances of the **vsftpd** daemon, a special `systemd` service unit (**vsftpd@.service**) for launching **vsftpd** as an instantiated service is supplied in the `vsftpd` package.

In order to make use of this service unit, a separate **vsftpd** configuration file for each required instance of the **FTP** server must be created and placed in the `/etc/vsftpd/` directory. Note that each of these configuration files must have a unique name (such as `/etc/vsftpd/vsftpd-site-2.conf`) and must be readable and writable only by the **root** user.

Within each configuration file for each **FTP** server listening on an **IPv4** network, the following directive must be unique:

```
listen_address=N.N.N.N
```

Replace `N.N.N.N` with a *unique IP* address for the **FTP** site being served. If the site is using **IPv6**, use the `listen_address6` directive instead.

Once there are multiple configuration files present in the `/etc/vsftpd/` directory, individual instances of the **vsftpd** daemon can be started by executing the following command as **root**:

```
~]# systemctl start vsftpd@configuration-file-name.service
```

In the above command, replace `configuration-file-name` with the unique name of the requested server's configuration file, such as `vsftpd-site-2`. Note that the configuration file's `.conf` extension should not be included in the command.

If you wish to start several instances of the **vsftpd** daemon at once, you can make use of a `systemd` target unit file (**vsftpd.target**), which is supplied in the `vsftpd` package. This `systemd` target causes an independent **vsftpd** daemon to be launched for each available **vsftpd** configuration file in the `/etc/vsftpd/` directory. Execute the following command as **root** to enable the target:

```
~]# systemctl enable vsftpd.target
ln -s '/usr/lib/systemd/system/vsftpd.target' '/etc/systemd/system/multi-user.target.wants/vsftpd.target'
```

The above command configures the systemd service manager to launch the **vsftpd** service (along with the configured **vsftpd** server instances) at boot time. To start the service immediately, without rebooting the system, execute the following command as **root**:

```
~]# systemctl start vsftpd.target
```

See [Section 6.3, “Working with systemd Targets”](#) for more information on how to use systemd targets to manage services.

Other directives to consider altering on a per-server basis are:

- ▶ **anon_root**
- ▶ **local_root**
- ▶ **vsftpd_log_file**
- ▶ **xferlog_file**

12.2.2.3. Encrypting vsftpd Connections Using SSL

In order to counter the inherently insecure nature of **FTP**, which transmits user names, passwords, and data without encryption by default, the **vsftpd** daemon can be configured to utilize the **SSL** or **TLS** protocols to authenticate connections and encrypt all transfers. Note that an **FTP** client that supports **SSL** is needed to communicate with **vsftpd** with **SSL** enabled.

Set the **ssl_enable** configuration directive in the **vsftpd.conf** file to **YES** to turn on **SSL** support. The default settings of other **SSL**-related directives that become automatically active when the **ssl_enable** option is enabled provide for a reasonably well-configured **SSL** set up. This includes, among other things, the requirement to use the **TLS v1** protocol for all connections or forcing all non-anonymous logins to use **SSL** for sending passwords and data transfers.

See the **vsftpd.conf(5)** manual page for other **SSL**-related configuration directives for fine-tuning the use of **SSL** by **vsftpd**.

12.2.2.4. SELinux Policy for vsftpd

The SELinux policy governing the **vsftpd** daemon (as well as other **ftpd** processes), defines a mandatory access control, which, by default, is based on least access required. In order to allow the **FTP** daemon to access specific files or directories, appropriate labels need to be assigned to them.

For example, in order to be able to share files anonymously, the **public_content_t** label must be assigned to the files and directories to be shared. You can do this using the **chcon** command as **root**:

```
~]# chcon -R -t public_content_t /path/to/directory
```

In the above command, replace */path/to/directory* with the path to the directory to which you wish to assign the label. Similarly, if you want to set up a directory for uploading files, you need to assign that particular directory the **public_content_rw_t** label. In addition to that, the **allow_ftpd_anon_write** SELinux Boolean option must be set to **1**. Use the **setsebool** command as **root** to do that:

```
~]# setsebool -P allow_ftpd_anon_write=1
```

If you want local users to be able to access their home directories through **FTP**, which is the default setting on Red Hat Enterprise Linux 7, the **ftp_home_dir** Boolean option needs to be set to **1**. If **vsftpd** is to be allowed to run in standalone mode, which is also enabled by default on Red Hat Enterprise Linux 7, the **ftpd_is_daemon** option needs to be set to **1** as well.

See the **ftpd_selinux(8)** manual page for more information, including examples of other useful labels and Boolean options, on how to configure the SELinux policy pertaining to **FTP**. Also, see the [Red Hat Enterprise Linux 7 SELinux User's and Administrator's Guide](#) for more detailed information about SELinux in general.

12.2.3. Additional Resources

For more information about **vsftpd**, see the following resources.

12.2.3.1. Installed Documentation

- ▶ The **/usr/share/doc/vsftpd-<version-number>/** directory — Replace *version-number* with the installed version of the **vsftpd** package. This directory contains a **README** file with basic information about the software. The **TUNING** file contains basic performance-tuning tips and the **SECURITY/** directory contains information about the security model employed by **vsftpd**.
- ▶ **vsftpd**-related manual pages — There are a number of manual pages for the daemon and the configuration files. The following lists some of the more important manual pages.

Server Applications

- **vsftpd(8)** — Describes available command-line options for **vsftpd**.

Configuration Files

- **vsftpd.conf(5)** — Contains a detailed list of options available within the configuration file for **vsftpd**.
- **hosts_access(5)** — Describes the format and options available within the **TCP** wrappers configuration files: **hosts.allow** and **hosts.deny**.

Interaction with SELinux

- **ftpd_selinux(8)** — Contains a description of the **SELinux** policy governing **ftpd** processes as well as an explanation of the way **SELinux** labels need to be assigned and Booleans set.

12.2.3.2. Online Documentation

About vsftpd and FTP in General

- ▶ <http://vsftpd.beasts.org/> — The **vsftpd** project page is a great place to locate the latest documentation and to contact the author of the software.
- ▶ <http://slacksite.com/other/ftp.html> — This website provides a concise explanation of the differences between active and passive-mode **FTP**.

Red Hat Enterprise Linux Documentation

- ▶ [Red Hat Enterprise Linux 7 Networking Guide](#) — The *Networking Guide* for Red Hat Enterprise Linux 7 documents relevant information regarding the configuration and administration of network interfaces, networks, and network services in this system. It provides an introduction to the **hostnamectl** utility and explains how to use it to view and set host

names on the command line, both locally and remotely.

- ▶ [Red Hat Enterprise Linux 7 SELinux User's and Administrator's Guide](#) — The *SELinux User's and Administrator's Guide* for Red Hat Enterprise Linux 7 describes the basic principles of **SELinux** and documents in detail how to configure and use **SELinux** with various services such as the **Apache HTTP Server**, **Postfix**, **PostgreSQL**, or **OpenShift**. It explains how to configure **SELinux** access permissions for system services managed by **systemd**.
- ▶ [Red Hat Enterprise Linux 7 Security Guide](#) — The *Security Guide* for Red Hat Enterprise Linux 7 assists users and administrators in learning the processes and practices of securing their workstations and servers against local and remote intrusion, exploitation, and malicious activity. It also explains how to secure critical system services.

Relevant RFC Documents

- ▶ [RFC 0959](#) — The original *Request for Comments* (RFC) of the **FTP** protocol from the IETF.
- ▶ [RFC 1123](#) — The small **FTP**-related section extends and clarifies RFC 0959.
- ▶ [RFC 2228](#) — **FTP** security extensions. **vsftpd** implements the small subset needed to support TLS and SSL connections.
- ▶ [RFC 2389](#) — Proposes **FEAT** and **OPTS** commands.
- ▶ [RFC 2428](#) — **IPv6** support.

12.3. Printer Configuration

The **Printer Configuration** tool serves for printer configuring, maintenance of printer configuration files, print spool directories and print filters, and printer classes management.

The tool is based on the Common Unix Printing System (CUPS). If you upgraded the system from a previous Red Hat Enterprise Linux version that used CUPS, the upgrade process preserved the configured printers.



Using the CUPS web application or command line tools

You can perform the same and additional operations on printers directly from the CUPS web application or command line. To access the application, in a web browser, go to <http://localhost:631/>. For CUPS manuals refer to the links on the **Home** tab of the web site.

12.3.1. Starting the Printer Configuration Tool

With the Printer Configuration tool you can perform various operations on existing printers and set up new printers. However, you can use also CUPS directly (go to <http://localhost:631/> to access CUPS).

On the panel, click **System** → **Administration** → **Printing**, or run the **system-config-printer** command from the command line to start the tool.

The **Printer Configuration** window depicted in [Figure 12.3, “Printer Configuration window”](#) appears.

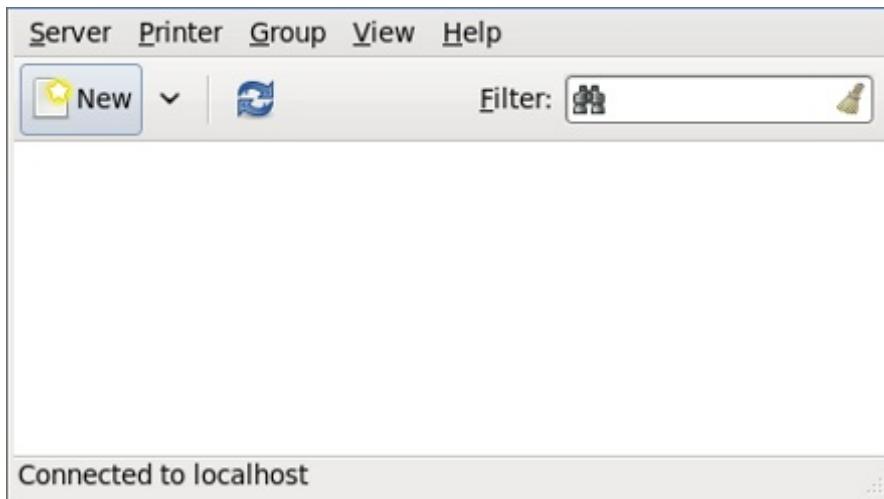


Figure 12.3. Printer Configuration window

12.3.2. Starting Printer Setup

Printer setup process varies depending on the printer queue type.

If you are setting up a local printer connected with USB, the printer is discovered and added automatically. You will be prompted to confirm the packages to be installed and provide the root password. Local printers connected with other port types and network printers need to be set up manually.

Follow this procedure to start a manual printer setup:

1. Start the Printer Configuration tool (refer to [Section 12.3.1, “Starting the Printer Configuration Tool”](#)).
2. Go to **Server** → **New** → **Printer**.
3. In the **Authenticate** dialog box, type the root user password and confirm.
4. Select the printer connection type and provide its details in the area on the right.

12.3.3. Adding a Local Printer

Follow this procedure to add a local printer connected with other than a serial port:

1. Open the **New Printer** dialog (refer to [Section 12.3.2, “Starting Printer Setup”](#)).
2. If the device does not appear automatically, select the port to which the printer is connected in the list on the left (such as **Serial Port #1** or **LPT #1**).
3. On the right, enter the connection properties:

for Other

URI (for example file:/dev/lp0)

for Serial Port

Baud Rate

Parity

Data Bits

Flow Control

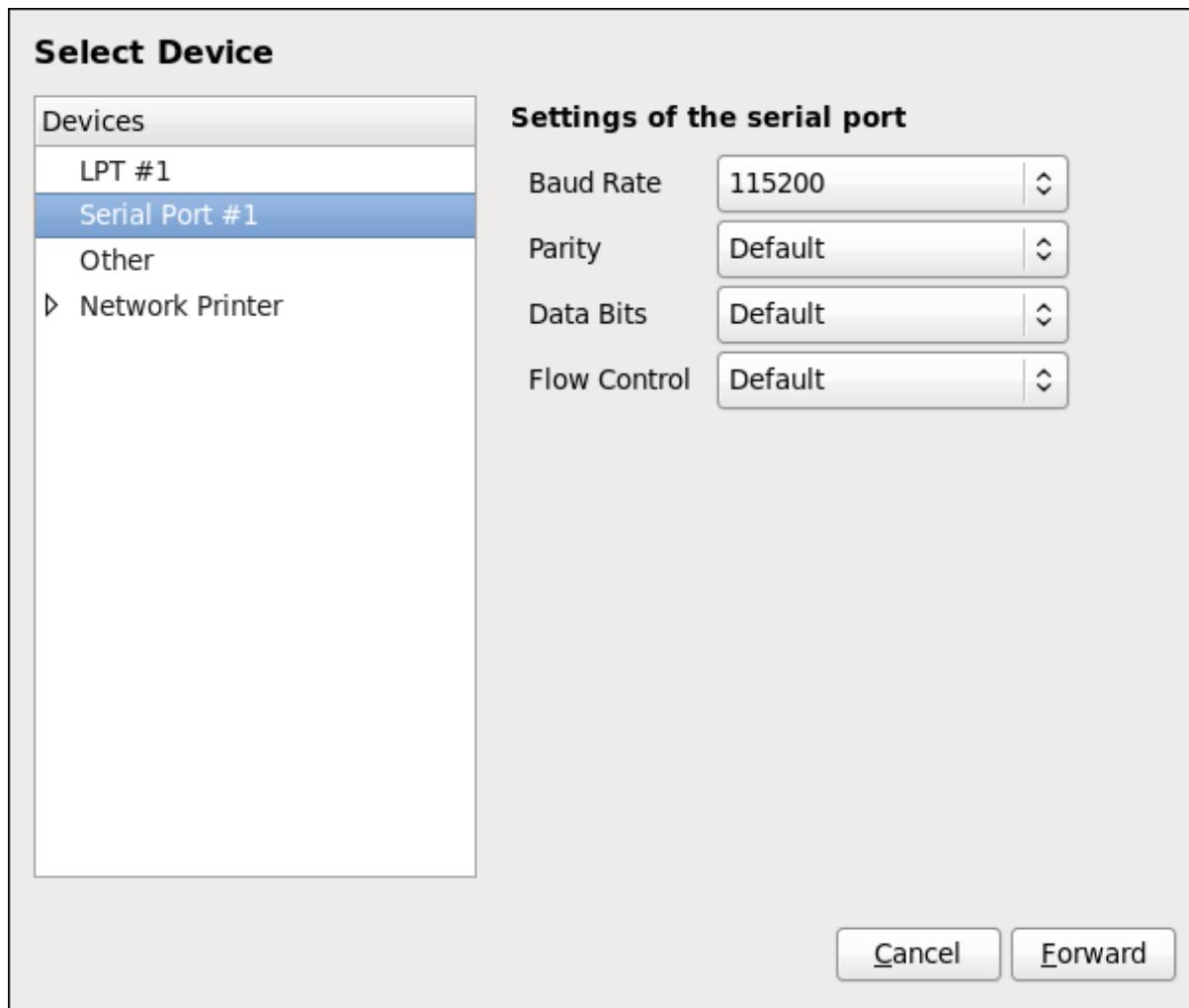


Figure 12.4. Adding a local printer

4. Click **Forward**.
5. Select the printer model. See [Section 12.3.8, “Selecting the Printer Model and Finishing”](#) for details.

12.3.4. Adding an AppSocket/HP JetDirect printer

Follow this procedure to add an AppSocket/HP JetDirect printer:

1. Open the **New Printer** dialog (refer to [Section 12.3.1, “Starting the Printer Configuration Tool”](#)).
2. In the list on the left, select **Network Printer** → **AppSocket/HP JetDirect**.
3. On the right, enter the connection settings:

Hostname

Printer hostname or IP address.

Port Number

Printer port listening for print jobs (**9100** by default).

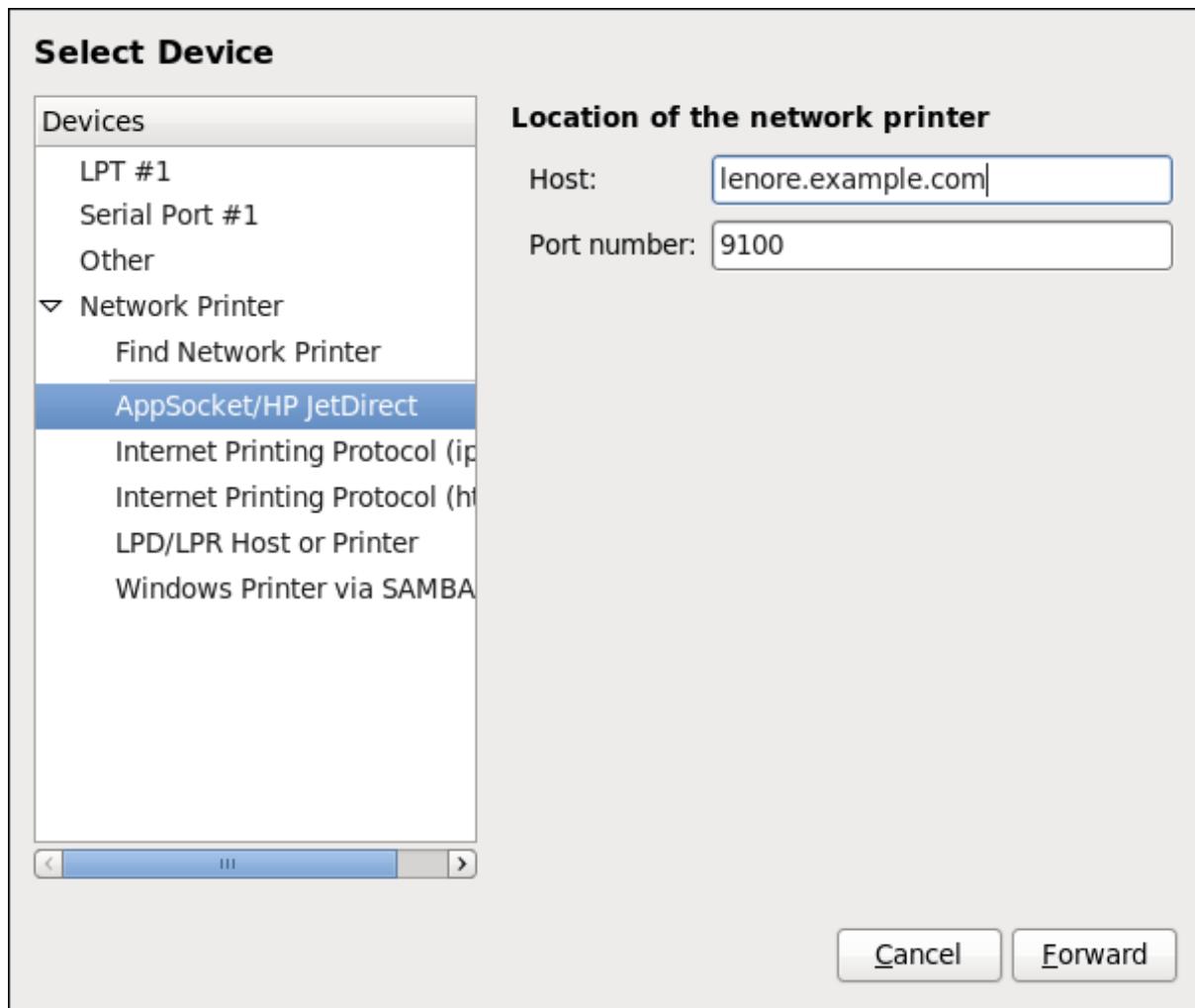


Figure 12.5. Adding a JetDirect printer

4. Click **Forward**.
5. Select the printer model. See [Section 12.3.8, “Selecting the Printer Model and Finishing”](#) for details.

12.3.5. Adding an IPP Printer

An IPP printer is a printer attached to a different system on the same TCP/IP network. The system this printer is attached to may either be running CUPS or simply configured to use IPP.

If a firewall is enabled on the printer server, then the firewall must be configured to allow incoming TCP connections on port 631. Note that the CUPS browsing protocol allows client machines to discover shared CUPS queues automatically. To enable this, the firewall on the client machine must be configured to allow incoming UDP packets on port 631.

Follow this procedure to add an IPP printer:

1. Open the **New Printer** dialog (refer to [Section 12.3.2, “Starting Printer Setup”](#)).
2. In the list of devices on the left, select **Network Printer** and **Internet Printing Protocol (ipp)** or **Internet Printing Protocol (https)**.

3. On the right, enter the connection settings:

Host

The hostname of the IPP printer.

Queue

The queue name to be given to the new queue (if the box is left empty, a name based on the device node will be used).

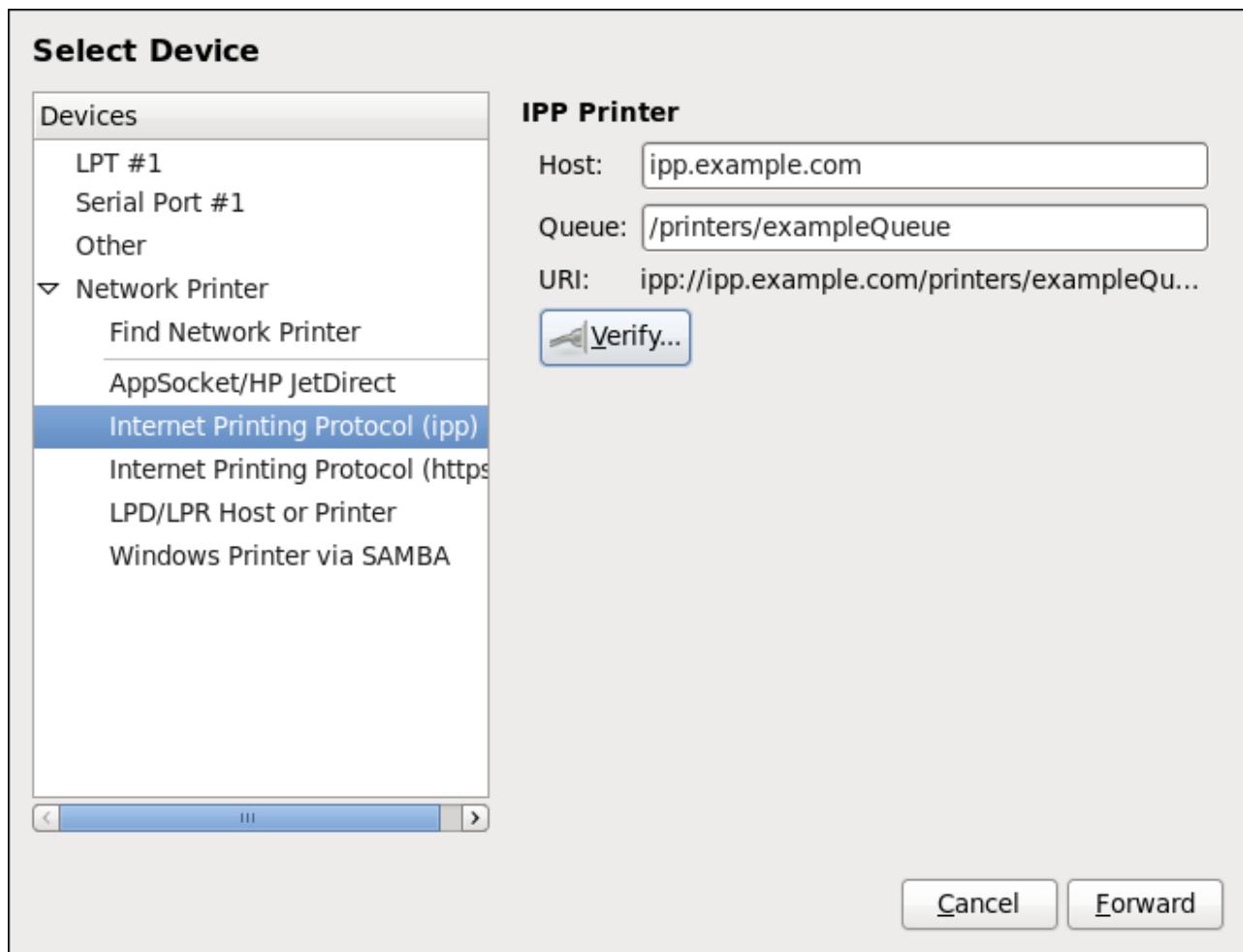


Figure 12.6. Adding an IPP printer

4. Click **Forward** to continue.
5. Select the printer model. See [Section 12.3.8, “Selecting the Printer Model and Finishing”](#) for details.

12.3.6. Adding an LPD/LPR Host or Printer

Follow this procedure to add an LPD/LPR host or printer:

1. Open the **New Printer** dialog (refer to [Section 12.3.2, “Starting Printer Setup”](#)).
2. In the list of devices on the left, select **Network Printer** → **LPD/LPR Host or Printer**.
3. On the right, enter the connection settings:

Host

The hostname of the LPD/LPR printer or host.

Optionally, click **Probe** to find queues on the LPD host.

Queue

The queue name to be given to the new queue (if the box is left empty, a name based on the device node will be used).

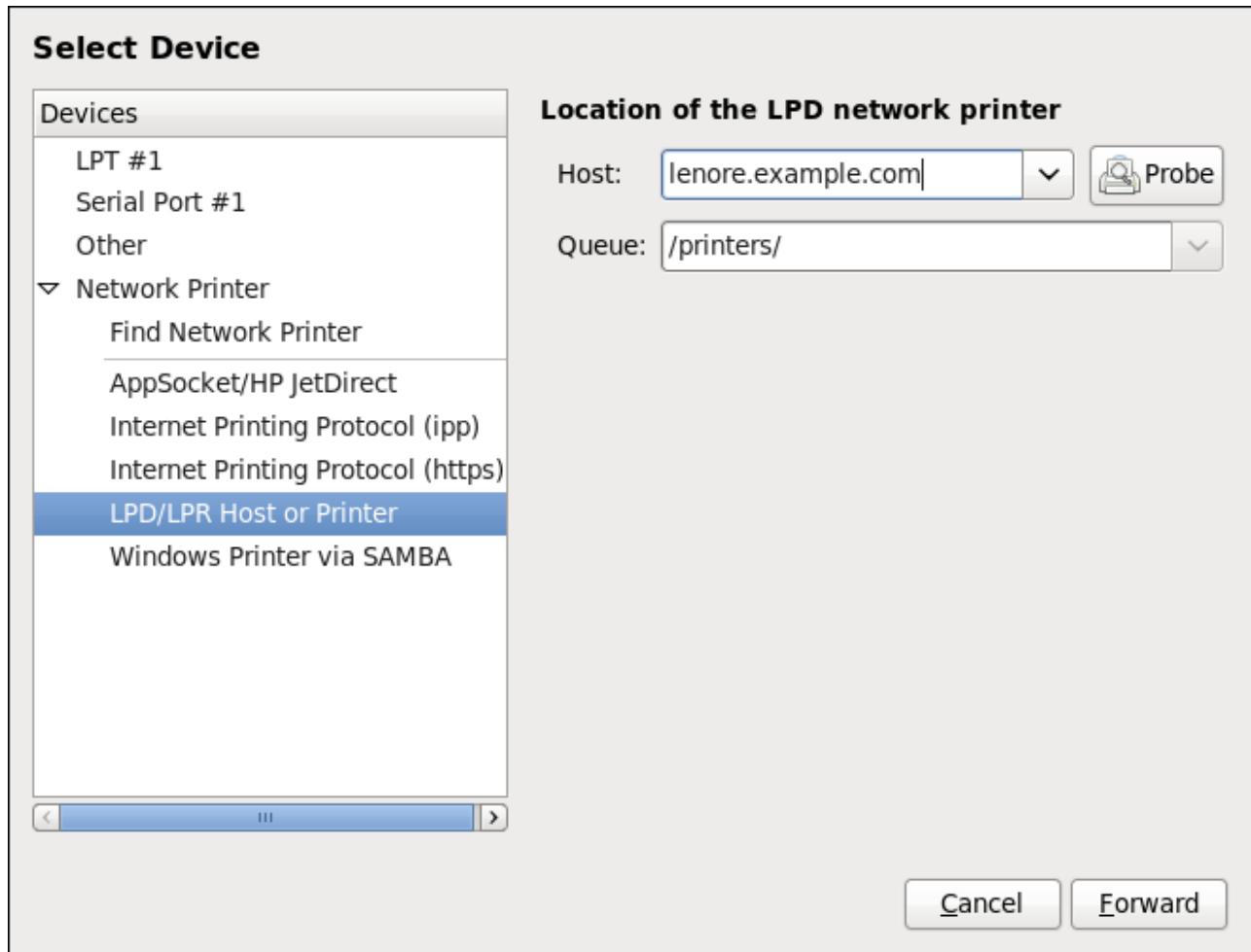


Figure 12.7. Adding an LPD/LPR printer

4. Click **Forward** to continue.
5. Select the printer model. See [Section 12.3.8, “Selecting the Printer Model and Finishing”](#) for details.

12.3.7. Adding a Samba (SMB) printer

Follow this procedure to add a Samba printer:

1. Open the **New Printer** dialog (refer to [Section 12.3.2, “Starting Printer Setup”](#)).
2. In the list on the left, select **Network Printer** → **Windows Printer via SAMBA**.

3. Enter the SMB address in the **smb://** field. Use the format *computer name/printer share*. In [Figure 12.8, “Adding a SMB printer”](#), the *computer name* is **dellbox** and the *printer share* is **r2**.

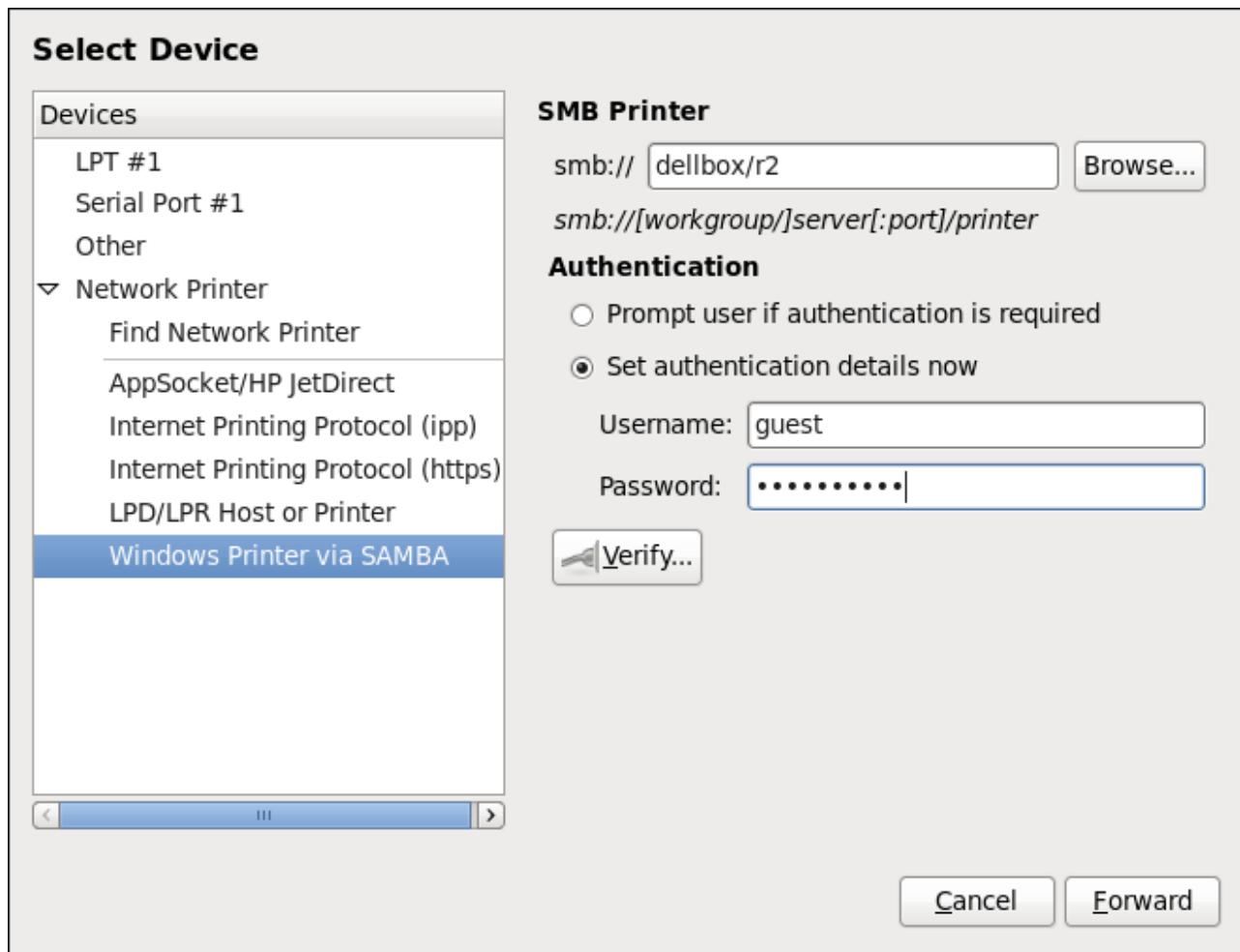


Figure 12.8. Adding a SMB printer

4. Click **Browse** to see the available workgroups/domains. To display only queues of a particular host, type in the host name (NetBios name) and click **Browse**.
5. Select either of the options:
 - A. **Prompt user if authentication is required**: username and password are collected from the user when printing a document.
 - B. **Set authentication details now**: provide authentication information now so it is not required later. In the **Username** field, enter the username to access the printer. This user must exist on the SMB system, and the user must have permission to access the printer. The default user name is typically **guest** for Windows servers, or **nobody** for Samba servers.
6. Enter the **Password** (if required) for the user specified in the **Username** field.



Be careful when choosing a password

Samba printer usernames and passwords are stored in the printer server as unencrypted files readable by root and lpd. Thus, other users that have root access to the printer server can view the username and password you use to access the Samba printer.

As such, when you choose a username and password to access a Samba printer, it is advisable that you choose a password that is different from what you use to access your local Red Hat Enterprise Linux system.

If there are files shared on the Samba print server, it is recommended that they also use a password different from what is used by the print queue.

7. Click **Verify** to test the connection. Upon successful verification, a dialog box appears confirming printer share accessibility.
8. Click **Forward**.
9. Select the printer model. See [Section 12.3.8, “Selecting the Printer Model and Finishing”](#) for details.

12.3.8. Selecting the Printer Model and Finishing

Once you have properly selected a printer connection type, the system attempts to acquire a driver. If the process fails, you can locate or search for the driver resources manually.

Follow this procedure to provide the printer driver and finish the installation:

1. In the window displayed after the automatic driver detection has failed, select one of the following options:
 - A. **Select a Printer from database** — the system chooses a driver based on the selected make of your printer from the list of **Makes**. If your printer model is not listed, choose **Generic**.
 - B. **Provide PPD file** — the system uses the provided PostScript Printer Description (PPD) file for installation. A PPD file may also be delivered with your printer as being normally provided by the manufacturer. If the PPD file is available, you can choose this option and use the browser bar below the option description to select the PPD file.
 - C. **Search for a printer driver to download** — enter the make and model of your printer into the **Make and model** field to search on OpenPrinting.org for the appropriate packages.

Choose Driver

- Select printer from database
- Provide PPD file
- Search for a printer driver to download

The foomatic printer database contains various manufacturer provided PostScript Printer Description (PPD) files and also can generate PPD files for a large number of (non PostScript) printers. But in general manufacturer provided PPD files provide better access to the specific features of the printer.

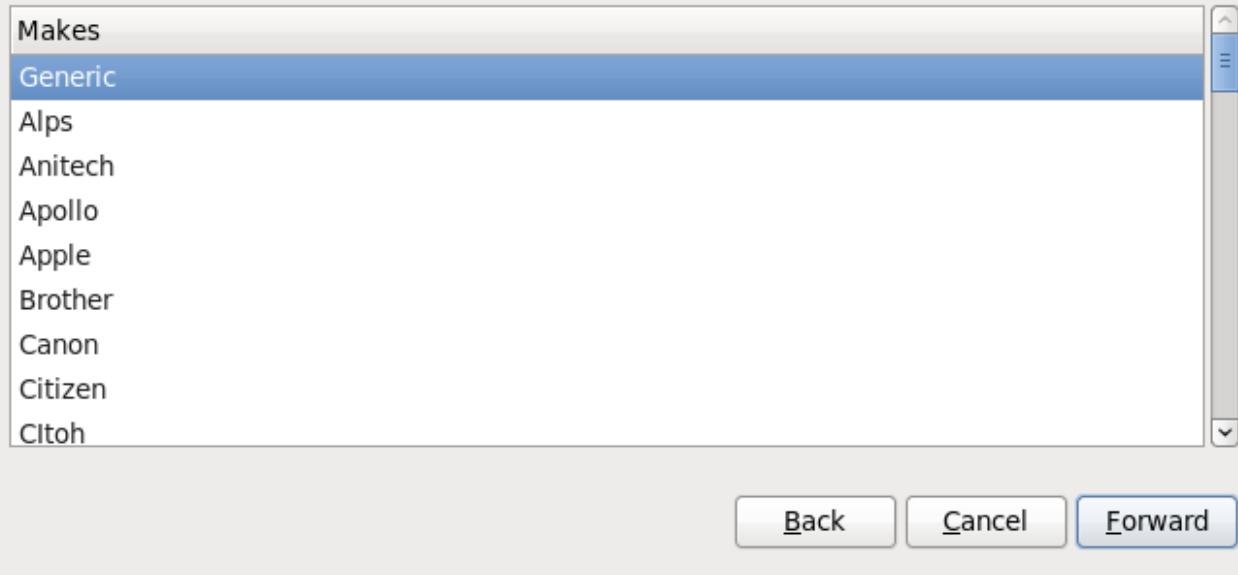


Figure 12.9. Selecting a printer brand

2. Depending on your previous choice provide details in the area displayed below:
 - » Printer brand for the **Select printer from database** option.
 - » PPD file location for the **Provide PPD file** option.
 - » Printer make and model for the **Search for a printer driver to download** option.
3. Click **Forward** to continue.
4. If applicable for your option, window shown in [Figure 12.10, "Selecting a printer model"](#) appears. Choose the corresponding model in the **Models** column on the left.

Selecting a printer driver

On the right, the recommended printed driver is automatically selected; however, you can select another available driver. The print driver processes the data that you want to print into a format the printer can understand. Since a local printer is attached directly to your computer, you need a printer driver to process the data that is sent to the printer.

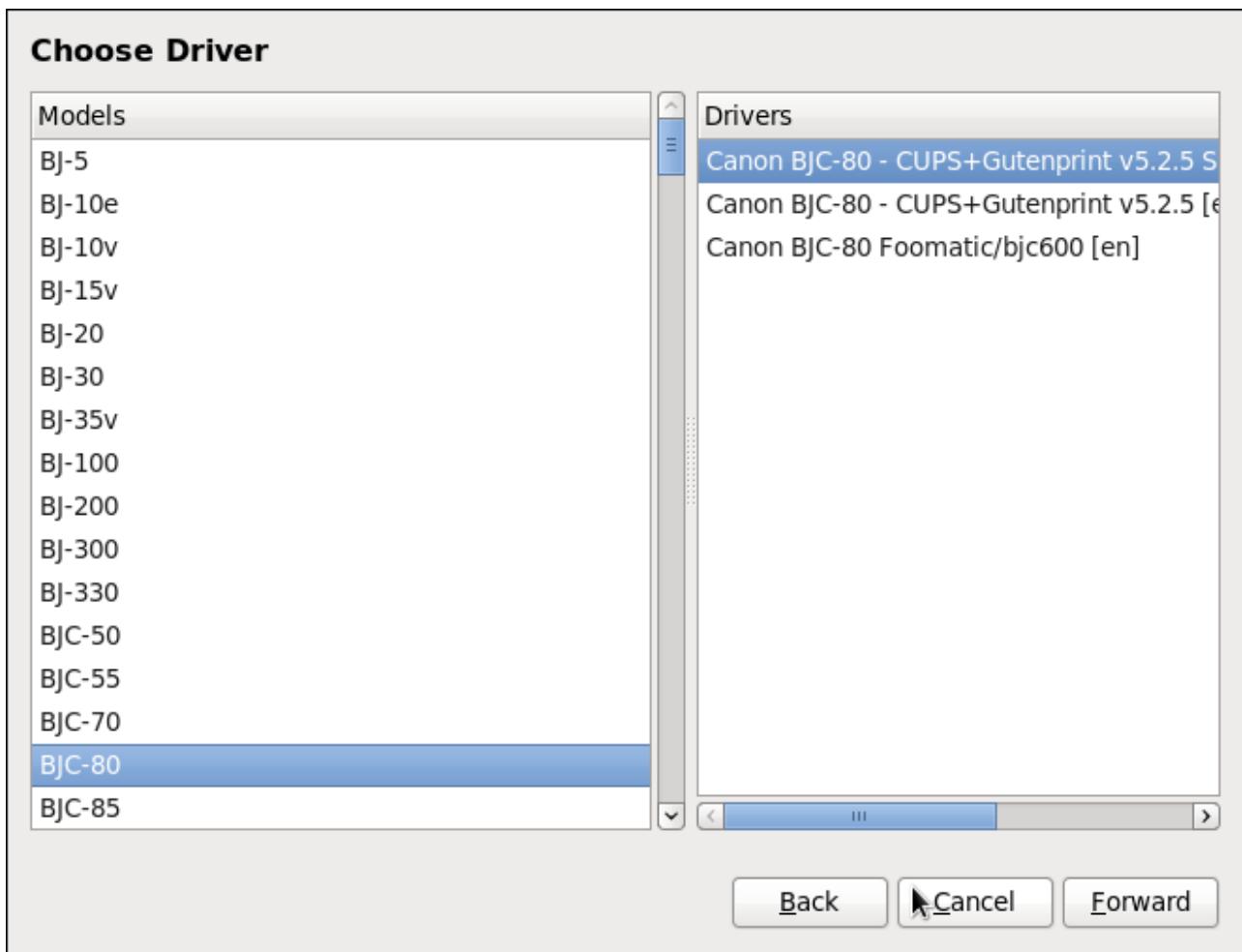


Figure 12.10. Selecting a printer model

5. Click **Forward**.
6. Under the **Describe Printer** enter a unique name for the printer in the **Printer Name** field. The printer name can contain letters, numbers, dashes (-), and underscores (_); it *must not* contain any spaces. You can also use the **Description** and **Location** fields to add further printer information. Both fields are optional, and may contain spaces.

Describe Printer

Printer Name
Short name for this printer such as "laserjet"

Description (optional)
Human-readable description such as "HP LaserJet with Duplexer"

Location (optional)
Human-readable location such as "Lab 1"

[Back](#) [Cancel](#) [Apply](#)

Figure 12.11. Printer setup

7. Click **Apply** to confirm your printer configuration and add the print queue if the settings are correct. Click **Back** to modify the printer configuration.
8. After the changes are applied, a dialog box appears allowing you to print a test page. Click **Yes** to print a test page now. Alternatively, you can print a test page later as described in [Section 12.3.9, "Printing a Test Page"](#).

12.3.9. Printing a Test Page

After you have set up a printer or changed a printer configuration, print a test page to make sure the printer is functioning properly:

1. Right-click the printer in the **Printing** window and click **Properties**.
2. In the Properties window, click **Settings** on the left.
3. On the displayed **Settings** tab, click the **Print Test Page** button.

12.3.10. Modifying Existing Printers

To delete an existing printer, in the **Printer Configuration** window, select the printer and go to **Printer → Delete**. Confirm the printer deletion. Alternatively, press the **Delete** key.

To set the default printer, right-click the printer in the printer list and click the **Set as Default** button in the context menu.

12.3.10.1. The Settings Page

To change printer driver configuration, double-click the corresponding name in the **Printer** list and click the **Settings** label on the left to display the **Settings** page.

You can modify printer settings such as make and model, print a test page, change the device location (URI), and more.

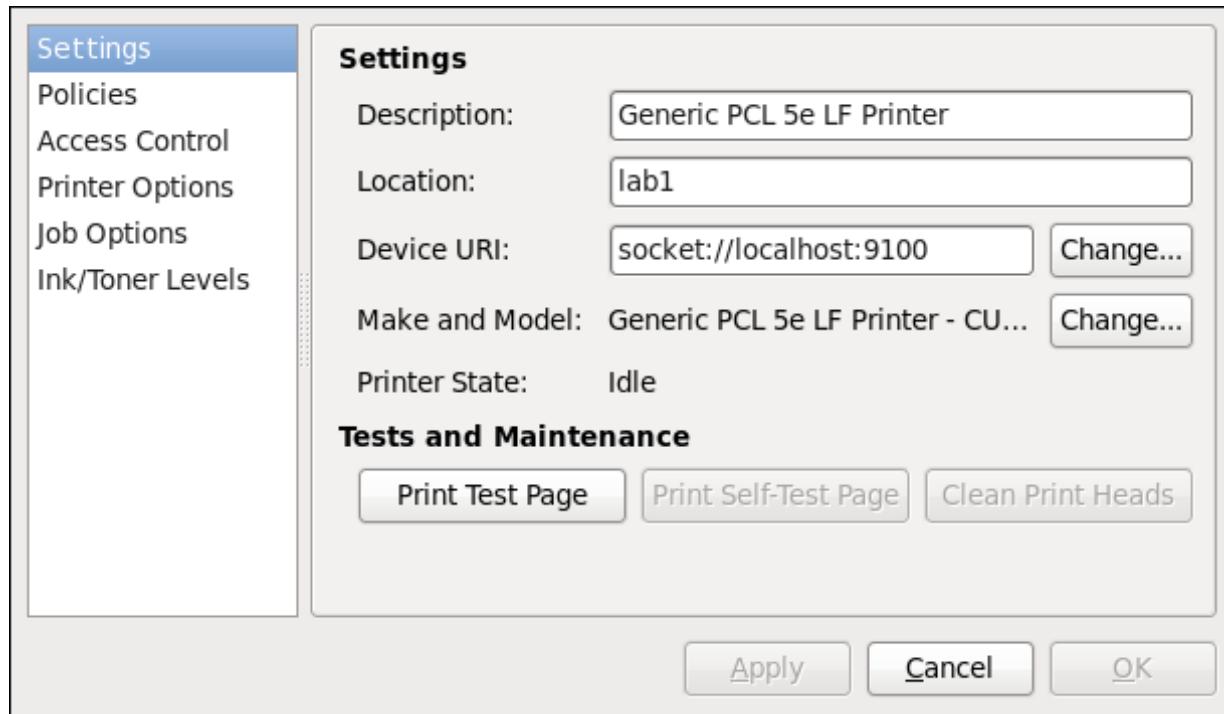


Figure 12.12. Settings page

12.3.10.2. The Policies Page

Click the **Policies** button on the left to change settings in printer state and print output.

You can select the printer states, configure the **Error Policy** of the printer (you can decide to abort the print job, retry, or stop it if an error occurs).

You can also create a *banner page* (a page that describes aspects of the print job such as the originating printer, the username from which the job originated, and the security status of the document being printed): click the **Starting Banner** or **Ending Banner** drop-menu and choose the option that best describes the nature of the print jobs (such as **topsecret**, **classified**, or **confidential**).

12.3.10.2.1. Sharing Printers

On the **Policies** page, you can mark a printer as shared: if a printer is shared, users published on the network can use it. To allow the sharing function for printers, go to **Server** → **Settings** and select **Publish shared printers connected to this system**.

Finally, make sure that the firewall allows incoming TCP connections to port 631 (that is Network Printing Server (IPP) in system-config-firewall).

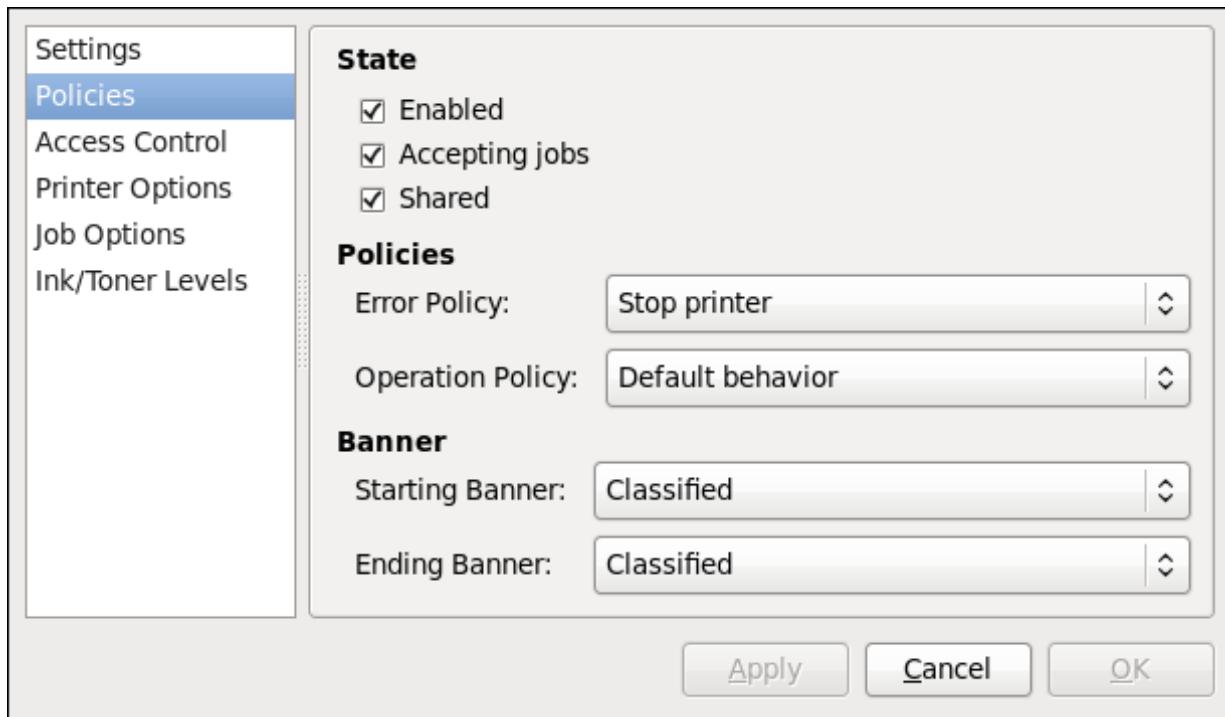


Figure 12.13. Policies page

12.3.10.2.2. The Access Control Page

You can change user-level access to the configured printer on the **Access Control** page. Click the **Access Control** label on the left to display the page. Select either **Allow printing for everyone except these users** or **Deny printing for everyone except these users** and define the user set below: enter the user name in the text box and click the **Add** button to add the user to the user set.

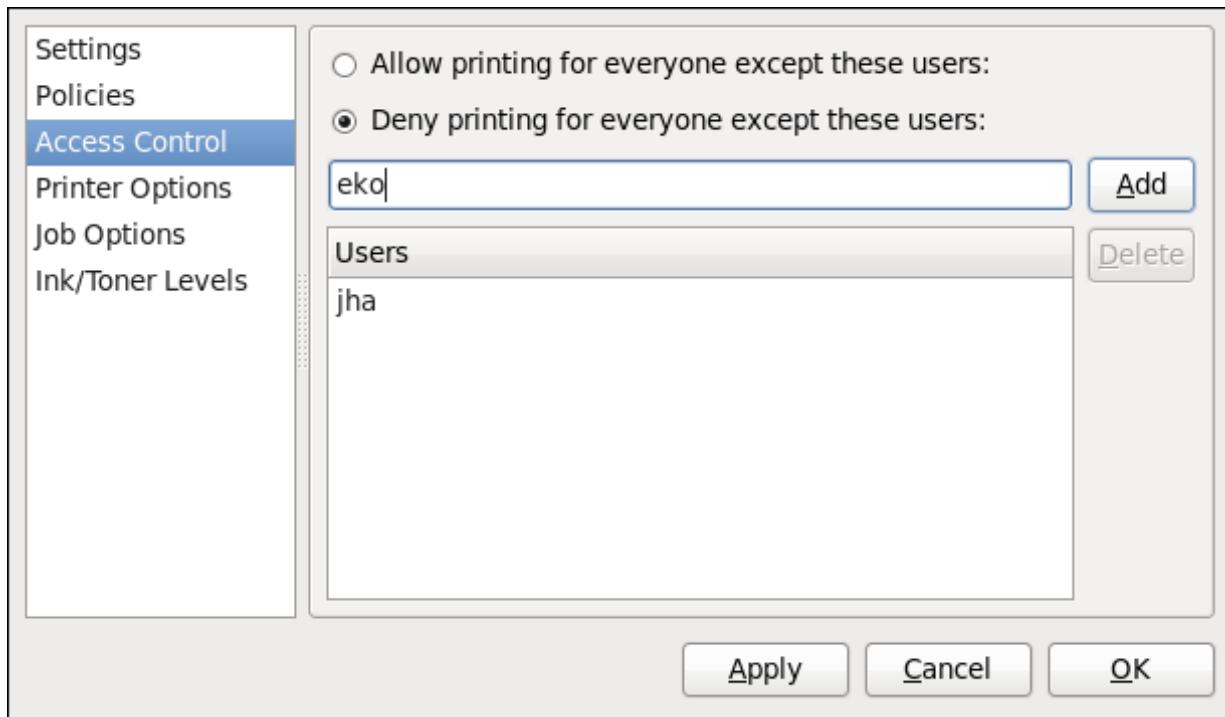


Figure 12.14. Access Control page

12.3.10.2.3. The Printer Options Page

The **Printer Options** page contains various configuration options for the printer media and output, and its content may vary from printer to printer. It contains general printing, paper, quality, and printing size settings.

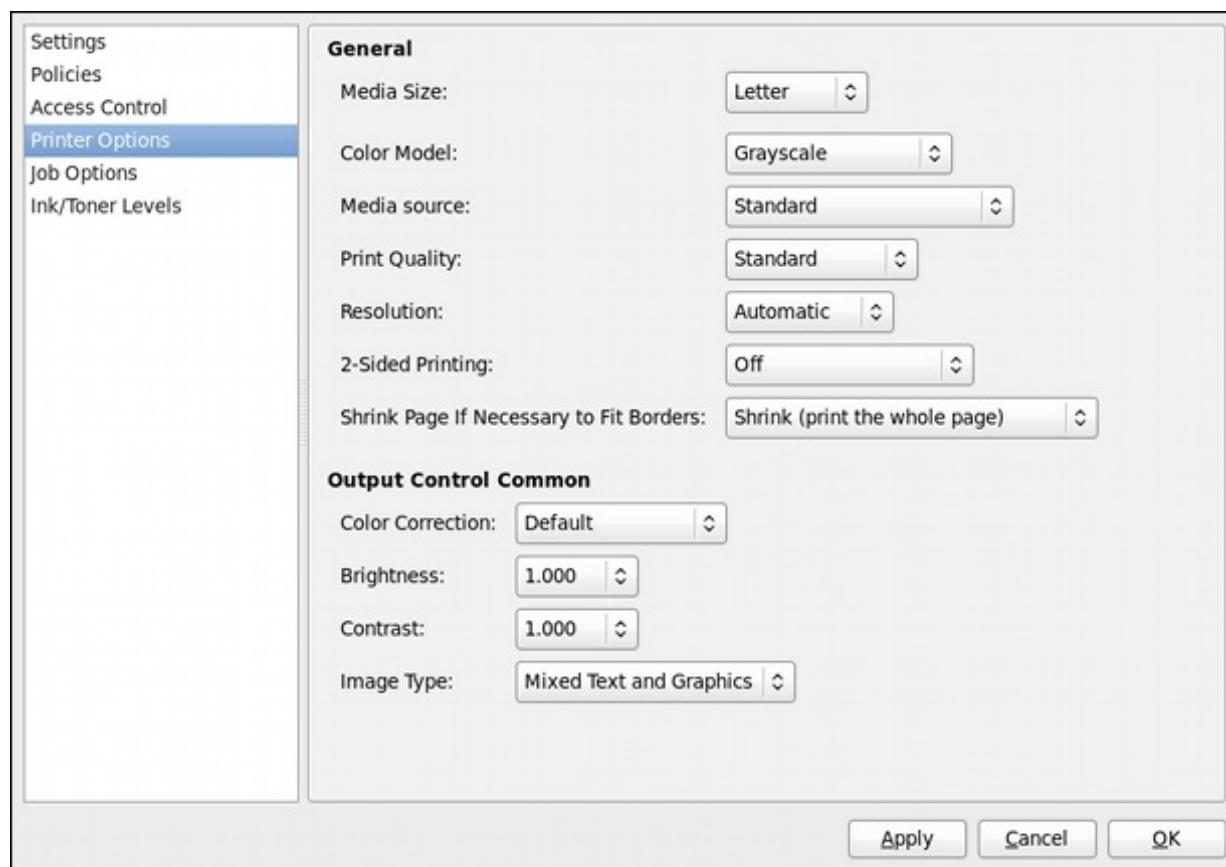


Figure 12.15. Printer Options page

12.3.10.2.4. Job Options Page

On the **Job Options** page, you can detail the printer job options. Click the **Job Options** label on the left to display the page. Edit the default settings to apply custom job options, such as number of copies, orientation, pages per side, scaling (increase or decrease the size of the printable area, which can be used to fit an oversize print area onto a smaller physical sheet of print medium), detailed text options, and custom job options.

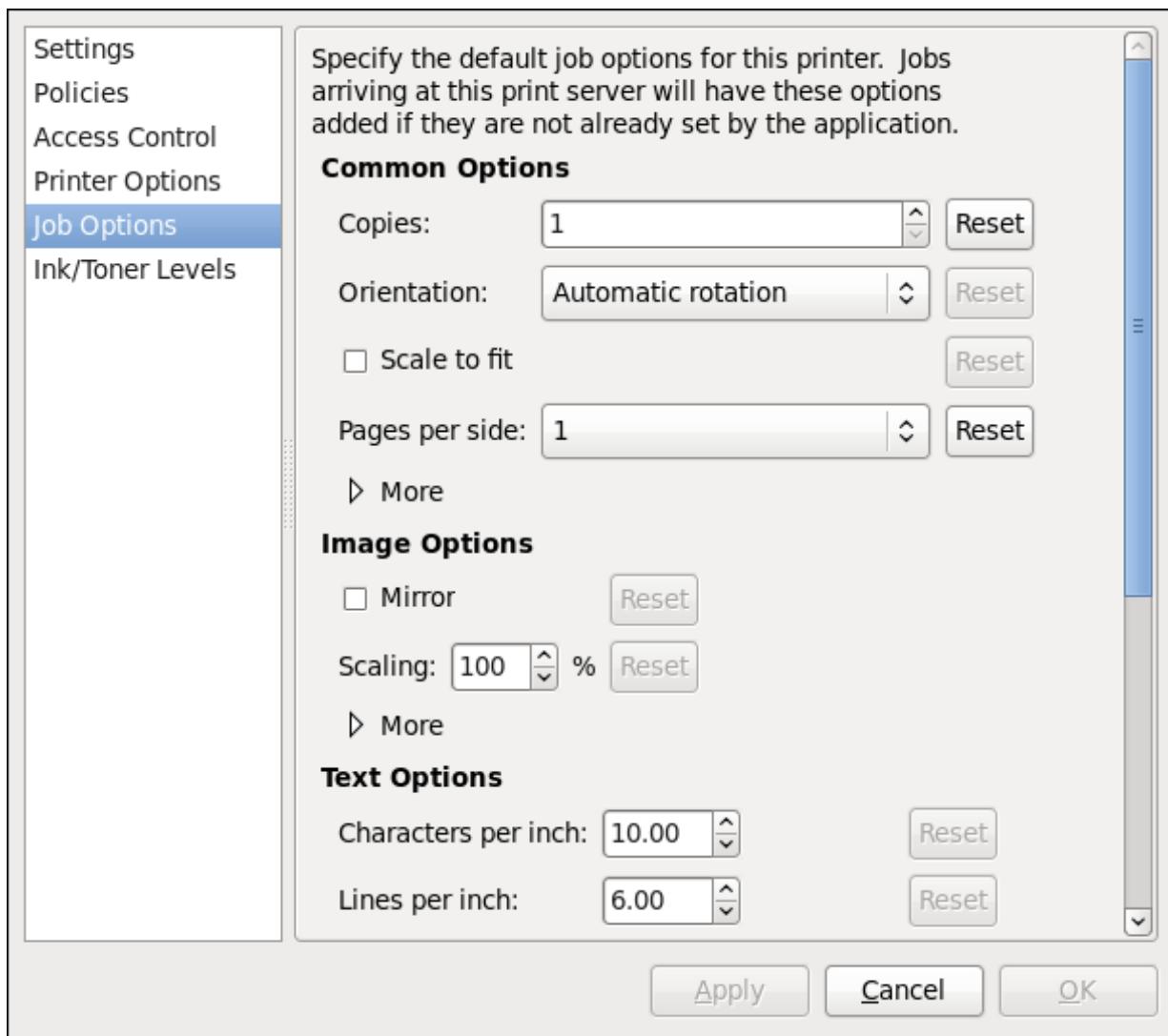


Figure 12.16. Job Options page

12.3.10.2.5. Ink/Toner Levels Page

The **Ink/Toner Levels** page contains details on toner status if available and printer status messages. Click the **Ink/Toner Levels** label on the left to display the page.

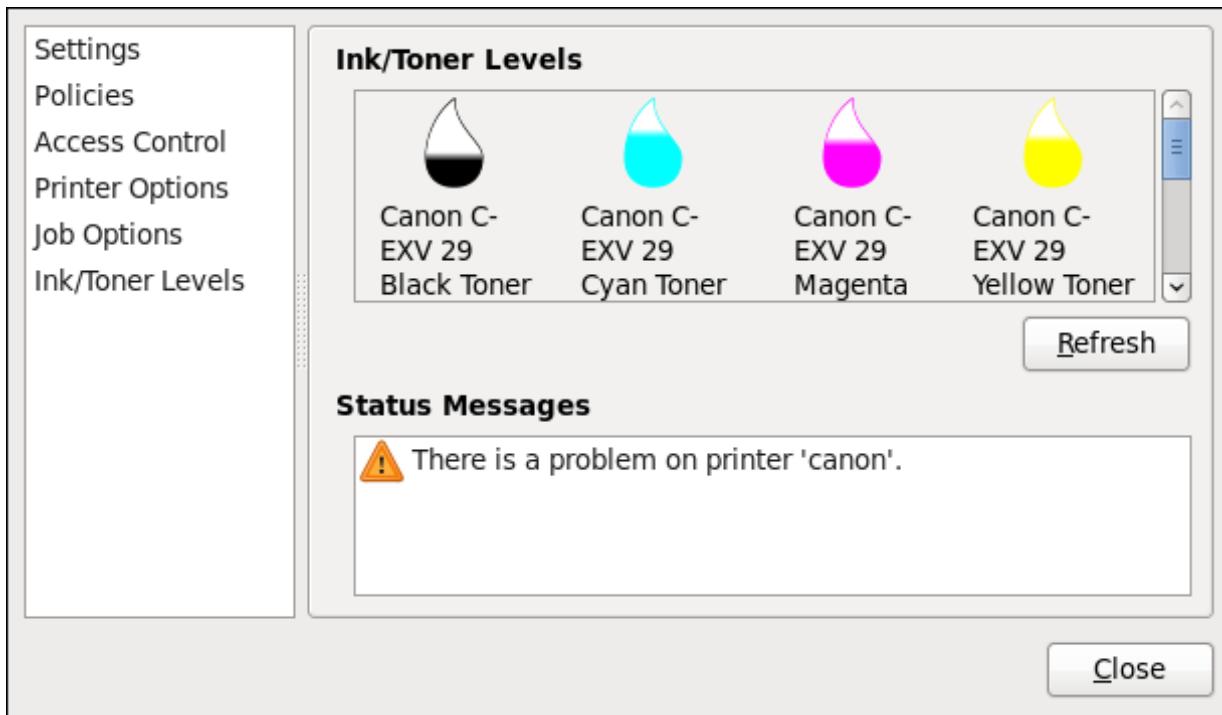


Figure 12.17. Ink/Toner Levels page

12.3.10.3. Managing Print Jobs

When you send a print job to the printer daemon, such as printing a text file from **Emacs** or printing an image from **GIMP**, the print job is added to the print spool queue. The print spool queue is a list of print jobs that have been sent to the printer and information about each print request, such as the status of the request, the job number, and more.

During the printing process, the **Printer Status** icon appears in the **Notification Area** on the panel. To check the status of a print job, click the **Printer Status**, which displays a window similar to [Figure 12.18, “GNOME Print Status”](#).

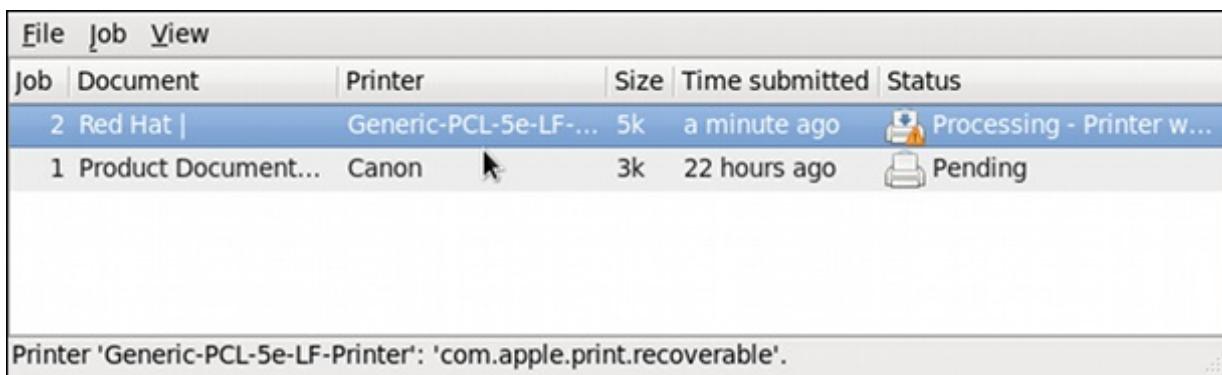


Figure 12.18. GNOME Print Status

To cancel, hold, release, reprint or authenticate a print job, select the job in the **GNOME Print Status** and on the **Job** menu, click the respective command.

To view the list of print jobs in the print spool from a shell prompt, type the command **lpstat -o**. The last few lines look similar to the following:

Example 12.1. Example of `lpstat -o` output

```
$ lpstat -o
Charlie-60          twaugh      1024  Tue 08 Feb 2011 16:42:11 GMT
Aaron-61           twaugh      1024  Tue 08 Feb 2011 16:42:44 GMT
Ben-62              root        1024  Tue 08 Feb 2011 16:45:42 GMT
```

If you want to cancel a print job, find the job number of the request with the command `lpstat -o` and then use the command `cancel job number`. For example, `cancel 60` would cancel the print job in [Example 12.1, “Example of `lpstat -o` output”](#). You can not cancel print jobs that were started by other users with the `cancel` command. However, you can enforce deletion of such job by issuing the `cancel -U root job_number` command. To prevent such canceling change the printer operation policy to **Authenticated** to force root authentication.

You can also print a file directly from a shell prompt. For example, the command `lp sample.txt` prints the text file `sample.txt`. The print filter determines what type of file it is and converts it into a format the printer can understand.

12.3.11. Additional Resources

To learn more about printing on Red Hat Enterprise Linux, see the following resources.

12.3.11.1. Installed Documentation

`man lp`

The manual page for the `lpr` command that allows you to print files from the command line.

`man cancel`

The manual page for the command line utility to remove print jobs from the print queue.

`man mpage`

The manual page for the command line utility to print multiple pages on one sheet of paper.

`man cupsd`

The manual page for the CUPS printer daemon.

`man cupsd.conf`

The manual page for the CUPS printer daemon configuration file.

`man classes.conf`

The manual page for the class configuration file for CUPS.

`man lpstat`

The manual page for the `lpstat` command, which displays status information about classes, jobs, and printers.

12.3.11.2. Useful Websites

<http://www.linuxprinting.org/>

GNU/Linux Printing contains a large amount of information about printing in Linux.

<http://www.cups.org/>

Documentation, FAQs, and newsgroups about CUPS.

Chapter 13. Configuring NTP Using the chrony Suite

Accurate time keeping is important for a number of reasons in IT. In networking for example, accurate time stamps in packets and logs are required. In Linux systems, the **NTP** protocol is implemented by a daemon running in user space.

The user space daemon updates the system clock running in the kernel. The system clock can keep time by using various clock sources. Usually, the *Time Stamp Counter* (TSC) is used. The TSC is a CPU register which counts the number of cycles since it was last reset. It is very fast, has a high resolution, and there are no interrupts.

There is a choice between the daemons **ntpd** and **chronyd**, which are available from the repositories in the *ntp* and *chrony* packages respectively. This section describes the use of the **chrony** suite of utilities to update the system clock on systems that do not fit into the conventional permanently networked, always on, dedicated server category.

13.1. Introduction to the chrony Suite

Chrony consists of **chronyd**, a daemon that runs in user space, and **chronyc**, a command line program for making adjustments to **chronyd**. Systems which are not permanently connected, or not permanently powered up, take a relatively long time to adjust their system clocks with **ntpd**. This is because many small corrections are made based on observations of the clocks drift and offset. Temperature changes, which may be significant when powering up a system, affect the stability of hardware clocks. Although adjustments begin within a few milliseconds of booting a system, acceptable accuracy may take anything from ten seconds from a warm restart to a number of hours depending on your requirements, operating environment and hardware. **chrony** is a different implementation of the **NTP** protocol than **ntpd**, it can adjust the system clock more rapidly.

13.1.1. Differences Between ntpd and chronyd

One of the main differences between **ntpd** and **chronyd** is in the algorithms used to control the computer's clock. Things **chronyd** can do better than **ntpd** are:

- ▶ **chronyd** can work well when external time references are only intermittently accessible whereas **ntpd** needs regular polling of time reference to work well.
- ▶ **chronyd** can perform well even when the network is congested for longer periods of time.
- ▶ **chronyd** can usually synchronize the clock faster and with better time accuracy.
- ▶ **chronyd** quickly adapts to sudden changes in the rate of the clock, for example, due to changes in the temperature of the crystal oscillator, whereas **ntpd** may need a long time to settle down again.
- ▶ **chronyd** in the default configuration never steps the time after the clock has been synchronized at system start, in order not to upset other running programs. **ntpd** can be configured to never step the time too, but it has to use a different means of adjusting the clock, which has some disadvantages.
- ▶ **chronyd** can adjust the rate of the clock on a Linux system in a larger range, which allows it to operate even on machines with a broken or unstable clock. For example, on some virtual machines.

Things **chronyd** can do that **ntpd** cannot do:

- ▶ **chrony** provides support for isolated networks where the only method of time correction is manual entry. For example, by the administrator looking at a clock. **chrony** can look at the errors corrected at different updates to estimate the rate at which the computer gains or loses time, and use this estimate to trim the computer clock subsequently.
- ▶ **chrony** provides support to work out the rate of gain or loss of the real-time clock, the hardware clock, that maintains the time when the computer is turned off. It can use this data when the system boots to set the system time using an adjusted value of the time taken from the real-time clock. This is, at time of writing, only available in Linux.

Things **ntpd** can do that **chrony** cannot do:

- ▶ **ntpd** fully supports **NTP** version 4 (*RFC 5905*), including broadcast, multicast, manycast clients and servers, and the orphan mode. It also supports extra authentication schemes based on public-key cryptography (*RFC 5906*). **chrony** uses **NTP** version 3 (*RFC 1305*), which is compatible with version 4.
- ▶ **ntpd** includes drivers for many reference clocks whereas **chrony** relies on other programs, for example **gpsd**, to access the data from the reference clocks.

13.1.2. Choosing Between NTP Daemons

- ▶ **Chrony** should be considered for all systems which are frequently suspended or otherwise intermittently disconnected and reconnected to a network. Mobile and virtual systems for example.
- ▶ The **NTP** daemon (**ntpd**) should be considered for systems which are normally kept permanently on. Systems which are required to use broadcast or multicast **IP**, or to perform authentication of packets with the **Autokey** protocol, should consider using **ntpd**. **Chrony** only supports symmetric key authentication using a message authentication code (MAC) with MD5, SHA1 or stronger hash functions, whereas **ntpd** also supports the **Autokey** authentication protocol which can make use of the PKI system. **Autokey** is described in *RFC 5906*.

13.2. Understanding chrony and Its Configuration

13.2.1. Understanding chronyd

The **chrony** daemon, **chronyd**, running in user space, makes adjustments to the system clock which is running in the kernel. It does this by consulting external time sources, using the **NTP** protocol, when ever network access allows it to do so. When external references are not available, **chronyd** will use the last calculated drift stored in the drift file. It can also be commanded manually to make corrections, by **chronyc**.

13.2.2. Understanding chronyc

The **chrony** daemon, **chronyd**, can be controlled by the command line utility **chronyc**. This utility provides a command prompt which allows entering of a number of commands to make changes to **chronyd**. The default configuration is for **chronyd** to only accept commands from a local instance of **chronyc**, but **chronyc** can be used to alter the configuration so that **chronyd** will allow external control. That is to say, **chronyc** can be run remotely after first configuring **chronyd** to accept remote connections. The **IP** addresses allowed to connect to **chronyd** should be tightly controlled.

13.2.3. Understanding the chrony Configuration Commands

The default configuration file for **chrony** is **/etc/chrony.conf**. The **-f** option can be used to specify an alternate configuration file path. See the **chronyd** man page for further options. For a complete list of the directives that can be used see <http://chrony.tuxfamily.org/manual.html#Configuration-file>. We present here a selection of configuration options:

Comments

Comments should be preceded by #, %, ; or !

allow

Optionally specify a host, subnet, or network from which to allow **NTP** connections to a machine acting as **NTP** server. The default is not to allow connections.

Examples:

1. `allow server1.example.com`

Use this form to specify a particular host, by its host name, to be allowed access.

2. `allow 192.0.2.0/24`

Use this form to specify a particular network to be allowed access.

3. `allow 2001:db8::/32`

Use this form to specify an **IPv6** address to be allowed access.

cmdallow

This is similar to the **allow** directive (see section **allow**), except that it allows control access (rather than **NTP** client access) to a particular subnet or host. (By “control access” is meant that **chronyc** can be run on those hosts and successfully connect to **chronyd** on this computer.) The syntax is identical. There is also a **cmddeny all** directive with similar behavior to the **cmdallow all** directive.

dumpdir

Path to the directory to save the measurement history across restarts of **chronyd** (assuming no changes are made to the system clock behavior whilst it is not running). If this capability is to be used (via the **dumponexit** command in the configuration file, or the **dump** command in **chronyc**), the **dumpdir** command should be used to define the directory where the measurement histories are saved.

dumponexit

If this command is present, it indicates that **chronyd** should save the measurement history for each of its time sources recorded whenever the program exits. (See the **dumpdir** command above).

local

The **local** keyword is used to allow **chronyd** to appear synchronized to real time (from the viewpoint of clients polling it), even if it has no current synchronization source. This option is normally used on computers in an isolated network, where several computers are required to synchronize to one other, this being the “master” which is kept vaguely in line with real time by manual input.

An example of the command is:

```
local stratum 10
```

A large value of 10 indicates that the clock is so many hops away from a reference clock that its time is fairly unreliable. Put another way, if the computer ever has access to another computer which is ultimately synchronized to a reference clock, it will almost certainly be at a stratum less than 10. Therefore, the choice of a high value like 10 for the **local** command prevents the machine's own time from ever being confused with real time, were it ever to leak out to clients that have visibility of real servers.

log

The **log** command indicates that certain information is to be logged. It accepts the following options:

measurements

This option logs the raw NTP measurements and related information to a file called **measurements.log**.

statistics

This option logs information about the regression processing to a file called **statistics.log**.

tracking

This option logs changes to the estimate of the system's gain or loss rate, and any slews made, to a file called **tracking.log**.

rtc

This option logs information about the system's real-time clock.

refclocks

This option logs the raw and filtered reference clock measurements to a file called **refclocks.log**.

tempcomp

This option logs the temperature measurements and system rate compensations to a file called **tempcomp.log**.

The log files are written to the directory specified by the **logdir** command. An example of the command is:

```
log measurements statistics tracking
```

logdir

This directive allows the directory where log files are written to be specified. An example of the use of this directive is:

```
logdir /var/log/chrony
```

makestep

Normally **chronyd** will cause the system to gradually correct any time offset, by slowing down or speeding up the clock as required. In certain situations, the system clock may be so far adrift that this slewing process would take a very long time to correct the system clock. This directive forces **chronyd** to step system clock if the adjustment is larger than a threshold value, but only if there were no more clock updates since **chronyd** was started than a specified limit (a negative value can be used to disable the limit). This is particularly useful when using reference clocks, because the **initstepslew** directive only works with **NTP** sources.

An example of the use of this directive is:

```
makestep 1000 10
```

This would step the system clock if the adjustment is larger than 1000 seconds, but only in the first ten clock updates.

maxchange

This directive sets the maximum allowed offset corrected on a clock update. The check is performed only after the specified number of updates to allow a large initial adjustment of the system clock. When an offset larger than the specified maximum occurs, it will be ignored for the specified number of times and then **chronyd** will give up and exit (a negative value can be used to never exit). In both cases a message is sent to syslog.

An example of the use of this directive is:

```
maxchange 1000 1 2
```

After the first clock update, **chronyd** will check the offset on every clock update, it will ignore two adjustments larger than 1000 seconds and exit on another one.

maxupdateskew

One of **chronyd**'s tasks is to work out how fast or slow the computer's clock runs relative to its reference sources. In addition, it computes an estimate of the error bounds around the estimated value. If the range of error is too large, it probably indicates that the measurements have not settled down yet, and that the estimated gain or loss rate is not very reliable. The **maxupdateskew** parameter allows the threshold for determining whether an estimate may be so unreliable that it should not be used. By default, the threshold is 1000 ppm. The format of the syntax is:

```
maxupdateskew skew-in-ppm
```

Typical values for *skew-in-ppm* might be 100 for a dial-up connection to servers over a telephone line, and 5 or 10 for a computer on a LAN. It should be noted that this is not the only means of protection against using unreliable estimates. At all times, **chronyd** keeps track of both the estimated gain or loss rate, and the error bound on the estimate. When a new estimate is generated following another measurement from one of the sources, a weighted combination algorithm is used to update the master estimate. So if **chronyd** has an existing highly-reliable master estimate and a new estimate is generated which has large error bounds, the existing master estimate will dominate in the new master estimate.

noclientlog

This directive, which takes no arguments, specifies that client accesses are not to be logged. Normally they are logged, allowing statistics to be reported using the clients command in **chronyc**.

reselectdist

When **chronyd** selects synchronization source from available sources, it will prefer the one with minimum synchronization distance. However, to avoid frequent reselecting when there are sources with similar distance, a fixed distance is added to the distance for sources that are currently not selected. This can be set with the **reselectdist** option. By default, the distance is 100 microseconds.

The format of the syntax is:

```
reselectdist dist-in-seconds
```

stratumweight

The **stratumweight** directive sets how much distance should be added per stratum to the synchronization distance when **chronyd** selects the synchronization source from available sources.

The format of the syntax is:

```
stratumweight dist-in-seconds
```

By default, *dist-in-seconds* is 1 second. This usually means that sources with lower stratum will be preferred to sources with higher stratum even when their distance is significantly worse. Setting **stratumweight** to 0 makes **chronyd** ignore stratum when selecting the source.

rtcfile

The **rtcfile** directive defines the name of the file in which **chronyd** can save parameters associated with tracking the accuracy of the system's real-time clock (RTC). The format of the syntax is:

```
rtcfile /var/lib/chrony/rtc
```

chronyd saves information in this file when it exits and when the **writertc** command is issued in **chronyc**. The information saved is the RTC's error at some epoch, that epoch (in seconds since January 1 1970), and the rate at which the RTC gains or loses time. Not all real-time clocks are supported as their code is system-specific. Note that if this directive is used then real-time clock should not be manually adjusted as this would interfere with **chrony**'s need to measure the rate at which the real-time clock drifts if it was adjusted at random intervals.

rtcsync

The **rtcsync** directive is present in the **/etc/chrony.conf** file by default. This will inform the kernel the system clock is kept synchronized and the kernel will update the real-time clock every 11 minutes.

13.2.4. Security with chronyc

As access to **chronyc** allows changing **chronyd** just as editing the configuration files would, access to **chronyc** should be limited. Passwords can be specified in the key file, written in ASCII or HEX, to restrict the use of **chronyc**. One of the entries is used to restrict the use of operational commands and is referred to as the command key. In the default configuration, a random command key is generated automatically on start. It should not be necessary to specify or alter it manually.

Other entries in the key file can be used as **NTP** keys to authenticate packets received from remote **NTP** servers or peers. The two sides need to share a key with identical ID, hash type and password in their key

file. This requires manually creating the keys and copying them over a secure medium, such as **SSH**. If the key ID was, for example, 10 then the systems that act as clients must have a line in their configuration files in the following format:

```
server w.x.y.z key 10
peer w.x.y.z key 10
```

The location of the key file is specified in the **/etc/chrony.conf** file. The default entry in the configuration file is:

```
keyfile /etc/chrony.keys
```

The command key number is specified in **/etc/chrony.conf** using the **commandkey** directive, it is the key **chronyd** will use for authentication of user commands. The directive in the configuration file takes the following form:

```
commandkey 1
```

An example of the format of the default entry in the key file, **/etc/chrony.keys**, for the command key is:

```
1 SHA1 HEX:A6CFC50C9C93AB6E5A19754C246242FC5471BCDF
```

Where **1** is the key ID, **SHA1** is the hash function to use, **HEX** is the format of the key, and **A6CFC50C9C93AB6E5A19754C246242FC5471BCDF** is the key randomly generated when **chronyd** was started for the first time. The key can be given in hexadecimal or ASCII format (the default).

A manual entry in the key file, used to authenticate packets from certain **NTP** servers or peers, can be as simple as the following:

```
20 foobar
```

Were **20** is the key ID and **foobar** is the secret authentication key. The default hash is MD5, and ASCII is the default format for the key.

By default, **chronyd** is configured to listen for commands only from **localhost (127.0.0.1 and ::1)** on port **323**. To access **chronyd** remotely with **chronyc**, any **bindcmdaddress** directives in the **/etc/chrony.conf** file should be removed to enable listening on all interfaces and the **cmdallow** directive should be used to allow commands from the remote **IP** address, network, or subnet. In addition, port **323** has to be opened in the firewall in order to connect from a remote system. Note that the **allow** directive is for **NTP** access whereas the **cmdallow** directive is to enable the receiving of remote commands. It is possible to make these changes temporarily using **chronyc** running locally. Edit the configuration file to make persistent changes.

The communication between **chronyc** and **chronyd** is done over **UDP**, so it needs to be authorized before issuing operational commands. To authorize, use the **authhash** and **password** commands as follows:

```
chronyc> authhash SHA1
chronyc> password HEX:A6CFC50C9C93AB6E5A19754C246242FC5471BCDF
200 OK
```

If **chronyc** is used to configure the local **chronyd**, the **-a** option will run the **authhash** and **password** commands automatically.

Only the following commands can be used without providing a password: **activity**, **authhash**, **dns**, **exit**, **help**, **password**, **quit**, **rtcdata**, **sources**, **sourcestats**, **tracking**, **waitsync**.

13.3. Using chrony

13.3.1. Checking if chrony is Installed

To check if **chrony** is installed, run the following command as **root**:

```
~]# yum install chrony
```

The default location for the **chrony** daemon is **/usr/sbin/chronyd**. The command line utility will be installed to **/usr/bin/chronyc**.

13.3.2. Installing chrony

To install **chrony**, run the following command as **root**:

```
~]# yum install chrony -y
```

The default location for the **chrony** daemon is **/usr/sbin/chronyd**. The command line utility will be installed to **/usr/bin/chronyc**.

13.3.3. Checking the Status of chronyd

To check the status of **chronyd**, issue the following command:

```
~]$ systemctl status chronyd
chronyd.service - NTP client/server
   Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled)
   Active: active (running) since Wed 2013-06-12 22:23:16 CEST; 11h ago
```

13.3.4. Starting chronyd

To start **chronyd**, issue the following command as **root**:

```
~]# systemctl start chronyd
```

To ensure **chronyd** starts automatically at system start, issue the following command as **root**:

```
~]# systemctl enable chronyd
```

13.3.5. Stopping chronyd

To stop **chronyd**, issue the following command as **root**:

```
~]# systemctl stop chronyd
```

To prevent **chronyd** from starting automatically at system start, issue the following command as **root**:

```
~]# systemctl disable chronyd
```

13.3.6. Checking if chrony is Synchronized

To check if **chrony** is synchronized, make use of the **tracking**, **sources**, and **sourcestats** commands.

13.3.6.1. Checking chrony Tracking

To check **chrony** tracking, issue the following command:

```
~]$ chronyc tracking
Reference ID      : 1.2.3.4 (a.b.c)
Stratum          : 3
Ref time (UTC)   : Fri Feb  3 15:00:29 2012
System time      : 0.000001501 seconds slow of NTP time
Last offset      : -0.000001632 seconds
RMS offset       : 0.000002360 seconds
Frequency        : 331.898 ppm fast
Residual freq    : 0.004 ppm
Skew             : 0.154 ppm
Root delay       : 0.373169 seconds
Root dispersion  : 0.024780 seconds
Update interval  : 64.2 seconds
Leap status      : Normal
```

The fields are as follows:

Reference ID

This is the reference ID and name (or IP address) if available, of the server to which the computer is currently synchronized. If this is **127.127.1.1** it means the computer is not synchronized to any external source and that you have the “local” mode operating (via the local command in **chronyc**, or the **local** directive in the **/etc/chrony.conf** file (see section **local**)).

Stratum

The stratum indicates how many hops away from a computer with an attached reference clock we are. Such a computer is a stratum-1 computer, so the computer in the example is two hops away (that is to say, a.b.c is a stratum-2 and is synchronized from a stratum-1).

Ref time

This is the time (UTC) at which the last measurement from the reference source was processed.

System time

In normal operation, **chronyd** never steps the system clock, because any jump in the timescale can have adverse consequences for certain application programs. Instead, any error in the system clock is corrected by slightly speeding up or slowing down the system clock until the error has been removed, and then returning to the system clock’s normal speed. A consequence of this is that there will be a period when the system clock (as read by other programs using the **gettimeofday()** system call, or by the date command in the shell) will be different from **chronyd**’s estimate of the current true time (which it reports to **NTP** clients when it is operating in server mode). The value reported on this line is the difference due to this effect.

Last offset

This is the estimated local offset on the last clock update.

RMS offset

This is a long-term average of the offset value.

Frequency

The “frequency” is the rate by which the system’s clock would be wrong if **chrony** was not correcting it. It is expressed in ppm (parts per million). For example, a value of 1ppm would mean that when the system’s clock thinks it has advanced 1 second, it has actually advanced by 1.000001 seconds relative to true time.

Residual freq

This shows the “residual frequency” for the currently selected reference source. This reflects any difference between what the measurements from the reference source indicate the frequency should be and the frequency currently being used. The reason this is not always zero is that a smoothing procedure is applied to the frequency. Each time a measurement from the reference source is obtained and a new residual frequency computed, the estimated accuracy of this residual is compared with the estimated accuracy (see **skew** next) of the existing frequency value. A weighted average is computed for the new frequency, with weights depending on these accuracies. If the measurements from the reference source follow a consistent trend, the residual will be driven to zero over time.

Skew

This is the estimated error bound on the frequency.

Root delay

This is the total of the network path delays to the stratum-1 computer from which the computer is ultimately synchronized. In certain extreme situations, this value can be negative. (This can arise in a symmetric peer arrangement where the computers’ frequencies are not tracking each other and the network delay is very short relative to the turn-around time at each computer.)

Root dispersion

This is the total dispersion accumulated through all the computers back to the stratum-1 computer from which the computer is ultimately synchronized. Dispersion is due to system clock resolution, statistical measurement variations etc.

Leap status

This is the leap status, which can be Normal, Insert second, Delete second or Not synchronized.

13.3.6.2. Checking chrony Sources

The sources command displays information about the current time sources that **chrony** is accessing. The optional argument -v can be specified, meaning verbose. In this case, extra caption lines are shown as a reminder of the meanings of the columns.

```
~]$ chronyc sources
 210 Number of sources = 3
 MS Name/IP address          Stratum Poll  Reach LastRx Last sample
=====
 #* GPS0                      0    4    377     11   -479ns[ -621ns] +/-  134ns
 ^? a.b.c                     2    6    377     23   -923us[ -924us] +/-   43ms
 ^+ d.e.f                     1    6    377     21  -2629us[ -2619us] +/-   86ms
```

The columns are as follows:

M

This indicates the mode of the source. ^ means a server, = means a peer and # indicates a locally connected reference clock.

S

This column indicates the state of the sources. "*" indicates the source to which **chrony** is currently synchronized. "+" indicates acceptable sources which are combined with the selected source. "-" indicates acceptable sources which are excluded by the combining algorithm. "?" indicates sources to which connectivity has been lost or whose packets do not pass all tests. "x" indicates a clock which **chrony** thinks is a falseticker (that is to say, its time is inconsistent with a majority of other sources). "~" indicates a source whose time appears to have too much variability. The "?" condition is also shown at start-up, until at least 3 samples have been gathered from it.

Name/IP address

This shows the name or the IP address of the source, or reference ID for reference clocks.

Stratum

This shows the stratum of the source, as reported in its most recently received sample. Stratum 1 indicates a computer with a locally attached reference clock. A computer that is synchronized to a stratum 1 computer is at stratum 2. A computer that is synchronized to a stratum 2 computer is at stratum 3, and so on.

Poll

This shows the rate at which the source is being polled, as a base-2 logarithm of the interval in seconds. Thus, a value of 6 would indicate that a measurement is being made every 64 seconds. **chrony** automatically varies the polling rate in response to prevailing conditions.

Reach

This shows the source's reachability register printed as octal number. The register has 8 bits and is updated on every received or missed packet from the source. A value of 377 indicates that a valid reply was received for all from the last eight transmissions.

LastRx

This column shows how long ago the last sample was received from the source. This is normally in seconds. The letters **m**, **h**, **d** or **y** indicate minutes, hours, days or years. A value of 10 years indicates there were no samples received from this source yet.

Last sample

This column shows the offset between the local clock and the source at the last measurement. The number in the square brackets shows the actual measured offset. This may be suffixed by **ns** (indicating nanoseconds), **us** (indicating microseconds), **ms** (indicating milliseconds), or **s** (indicating seconds). The number to the left of the square brackets shows the original measurement, adjusted to allow for any slews applied to the local clock since. The number following the +/- indicator shows the margin of error in the measurement. Positive offsets indicate that the local clock is fast of the source.

13.3.6.3. Checking chrony Source Statistics

The **sourcestats** command displays information about the drift rate and offset estimation process for each of the sources currently being examined by **chronyd**. The optional argument **-v** can be specified, meaning verbose. In this case, extra caption lines are shown as a reminder of the meanings of the columns.

```
~]$ chronyc sourcestats
```

Name/IP Address	NP	NR	Span	Frequency	Freq	Skew	Offset	Std Dev
abc.def.ghi								

The columns are as follows:

Name/IP address

This is the name or **IP** address of the **NTP** server (or peer) or reference ID of the reference clock to which the rest of the line relates.

NP

This is the number of sample points currently being retained for the server. The drift rate and current offset are estimated by performing a linear regression through these points.

NR

This is the number of runs of residuals having the same sign following the last regression. If this number starts to become too small relative to the number of samples, it indicates that a straight line is no longer a good fit to the data. If the number of runs is too low, **chronyd** discards older samples and re-runs the regression until the number of runs becomes acceptable.

Span

This is the interval between the oldest and newest samples. If no unit is shown the value is in seconds. In the example, the interval is 46 minutes.

Frequency

This is the estimated residual frequency for the server, in parts per million. In this case, the computer's clock is estimated to be running 1 part in 10^9 slow relative to the server.

Freq Skew

This is the estimated error bounds on Freq (again in parts per million).

Offset

This is the estimated offset of the source.

Std Dev

This is the estimated sample standard deviation.

13.3.7. Manually Adjusting the System Clock

To update, or step, the system clock immediately, bypassing any adjustments in progress by slewing the clock, issue the following commands as **root**:

```
~]# chronyc
chrony> password commandkey-password
200 OK
chrony> makestep
200 OK
```

Where *commandkey-password* is the command key or password stored in the key file.

The real-time clock should not be manually adjusted if the **rtcfiile** directive is used as this would interfere with **chrony**'s need to measure the rate at which the real-time clock drifts if it was adjusted at random intervals.

If **chronyc** is used to configure the local **chronyd**, the **-a** will run the **authhash** and **password** commands automatically. This means that the interactive session illustrated above can be replaced by:

```
chronyc -a makestep
```

13.4. Setting Up chrony for Different Environments

13.4.1. Setting Up chrony for a System Which is Infrequently Connected

This example is intended for systems which use dial-on-demand connections. The normal configuration should be sufficient for mobile and virtual devices which connect intermittently. First, review and confirm that the default settings in the **/etc/chrony.conf** are similar to the following:

```
driftfile /var/lib/chrony/drift
commandkey 1
keyfile /etc/chrony.keys
```

The command key ID is generated at install time and should correspond with the **commandkey** value in the key file, **/etc/chrony.keys**.

1. Using your editor running as **root**, add the addresses of four **NTP** servers as follows:

```
server 0.pool.ntp.org offline
server 1.pool.ntp.org offline
server 2.pool.ntp.org offline
server 3.pool.ntp.org offline
```

The **offline** option can be useful in preventing systems from trying to activate connections. The **chrony** daemon will wait for **chronyc** to inform it that the system is connected to the network or Internet.

13.4.2. Setting Up chrony for a System in an Isolated Network

For a network that is never connected to the Internet, one computer is selected to be the master timeserver. The other computers are either direct clients of the master, or clients of clients. On the master, the drift file must be manually set with the average rate of drift of the system clock. If the master is rebooted it will obtain the time from surrounding systems and take an average to set its system clock. Thereafter it resumes applying adjustments based on the drift file. The drift file will be updated automatically when the **settime** command is used.

On the system selected to be the master, using a text editor running as **root**, edit the **/etc/chrony.conf** as follows:

```
driftfile /var/lib/chrony/drift
commandkey 1
keyfile /etc/chrony.keys
initstepslew 10 client1 client3 client6
local stratum 8
manual
allow 192.0.2.0
```

Where **192.0.2.0** is the network or subnet address from which the clients are allowed to connect.

On the systems selected to be direct clients of the master, using a text editor running as **root**, edit the **/etc/chrony.conf** as follows:

```
server master
driftfile /var/lib/chrony/drift
logdir /var/log/chrony
log measurements statistics tracking
keyfile /etc/chrony.keys
commandkey 24
local stratum 10
initstepslew 20 master
allow 192.0.2.123
```

Where **192.0.2.123** is the address of the master, and **master** is the host name of the master. Clients with this configuration will resynchronize the master if it restarts.

On the client systems which are not to be direct clients of the master, the **/etc/chrony.conf** file should be the same except that the **local** and **allow** directives should be omitted.

13.5. Using chronyc

13.5.1. Using chronyc to Control chronyd

To make changes using the command line utility **chronyc** in interactive mode, enter the following command as **root**:

```
~]# chronyc
```

chronyc must run as **root** if some of the restricted commands are to be used.

The **chronyc** command prompt will be displayed as follows:

```
chronyc>
```

You can type **help** to list all of the commands.

The utility can also be invoked in non-interactive command mode if called together with a command as follows:

```
~]# chronyc command
```

13.5.2. Using chronyc for Remote Administration

To configure **chrony** to connect to a remote instance of **chronyd**, issue a command as **root** in the following format:

```
~]# chronyc -h hostname
```

Where *hostname* is the **hostname** of a system running **chronyd** to connect to in order to allow remote administration from that host. The default is to connect to the daemon on the localhost.

To configure **chrony** to connect to a remote instance of **chronyd** on a non-default port, issue a command as **root** in the following format:

```
~]# chronyc -h hostname -p port
```

Where *port* is the port in use for controlling and monitoring by the instance of **chronyd** to be connected to.

Note that commands issued at the **chrony** command prompt are not persistent. Only commands in the configuration file are persistent.

From the remote systems, the system administrator can issue commands after first using the **password** command, preceded by the **authhash** command if the key used a hash different from MD5, at the **chronyc** command prompt as follows:

```
chronyc> password secretpasswordwithnospaces
200 OK
```

The password or hash associated with the command key for a remote system is best obtained by **SSH**. That is to say, an **SSH** connection should be established to the remote machine and the ID of the command key from **/etc/chrony.conf** and the command key in **/etc/chrony.keys** memorized or stored securely for the duration of the session.

13.6. Additional Resources

The following sources of information provide additional resources regarding **chrony**.

13.6.1. Installed Documentation

- ▶ **chrony(1)** man page — Introduces the **chrony** daemon and the command-line interface tool.
- ▶ **chronyc(1)** man page — Describes the **chronyc** command-line interface tool including commands and command options.
- ▶ **chronyd(1)** man page — Describes the **chronyd** daemon including commands and command options.
- ▶ **chrony.conf(5)** man page — Describes the **chrony** configuration file.
- ▶ **/usr/share/doc/chrony*/chrony.txt** — User guide for the **chrony** suite.

13.6.2. Useful Websites

<http://chrony.tuxfamily.org/manual.html>

The on-line user guide for **chrony**.

Chapter 14. Configuring NTP Using ntpd

14.1. Introduction to NTP

The *Network Time Protocol* (NTP) enables the accurate dissemination of time and date information in order to keep the time clocks on networked computer systems synchronized to a common reference over the network or the Internet. Many standards bodies around the world have atomic clocks which may be made available as a reference. The satellites that make up the Global Position System contain more than one atomic clock, making their time signals potentially very accurate. Their signals can be deliberately degraded for military reasons. An ideal situation would be where each site has a server, with its own reference clock attached, to act as a site-wide time server. Many devices which obtain the time and date via low frequency radio transmissions or the Global Position System (GPS) exist. However for most situations, a range of publicly accessible time servers connected to the Internet at geographically dispersed locations can be used. These **NTP** servers provide “*Coordinated Universal Time*” (UTC). Information about these time servers can be found at www.pool.ntp.org.

Accurate time keeping is important for a number of reasons in IT. In networking for example, accurate time stamps in packets and logs are required. Logs are used to investigate service and security issues and so timestamps made on different systems must be made by synchronized clocks to be of real value. As systems and networks become increasingly faster, there is a corresponding need for clocks with greater accuracy and resolution. In some countries there are legal obligations to keep accurately synchronized clocks. Please see www.ntp.org for more information. In Linux systems, **NTP** is implemented by a daemon running in user space. The default **NTP** user space daemon in Red Hat Enterprise Linux 7 is **chrony**. It must be disabled if you want to use the **ntpd** daemon. See [Chapter 13, Configuring NTP Using the chrony Suite](#) for information on **chrony**.

The user space daemon updates the system clock, which is a software clock running in the kernel. Linux uses a software clock as its system clock for better resolution than the typical embedded hardware clock referred to as the “*Real Time Clock*” (RTC). See the **rtc(4)** and **hwclock(8)** man pages for information on hardware clocks. The system clock can keep time by using various clock sources. Usually, the *Time Stamp Counter* (TSC) is used. The TSC is a CPU register which counts the number of cycles since it was last reset. It is very fast, has a high resolution, and there are no interrupts. On system start, the system clock reads the time and date from the RTC. The time kept by the RTC will drift away from actual time by up to 5 minutes per month due to temperature variations. Hence the need for the system clock to be constantly synchronized with external time references. When the system clock is being synchronized by **ntpd**, the kernel will in turn update the RTC every 11 minutes automatically.

14.2. NTP Strata

NTP servers are classified according to their synchronization distance from the atomic clocks which are the source of the time signals. The servers are thought of as being arranged in layers, or strata, from 1 at the top down to 15. Hence the word stratum is used when referring to a specific layer. Atomic clocks are referred to as Stratum 0 as this is the source, but no Stratum 0 packet is sent on the Internet, all stratum 0 atomic clocks are attached to a server which is referred to as stratum 1. These servers send out packets marked as Stratum 1. A server which is synchronized by means of packets marked stratum **n** belongs to the next, lower, stratum and will mark its packets as stratum **n+1**. Servers of the same stratum can exchange packets with each other but are still designated as belonging to just the one stratum, the stratum one below the best reference they are synchronized to. The designation Stratum 16 is used to indicate that the server is not currently synchronized to a reliable time source.

Note that by default **NTP** clients act as servers for those systems in the stratum below them.

Here is a summary of the **NTP** Strata:

Stratum 0:

Atomic Clocks and their signals broadcast over Radio and GPS

- ▶ GPS (Global Positioning System)
- ▶ Mobile Phone Systems
- ▶ Low Frequency Radio Broadcasts WWVB (Colorado, USA.), JJY-40 and JJY-60 (Japan), DCF77 (Germany), and MSF (United Kingdom)

These signals can be received by dedicated devices and are usually connected by RS-232 to a system used as an organizational or site-wide time server.

Stratum 1:

Computer with radio clock, GPS clock, or atomic clock attached

Stratum 2:

Reads from stratum 1; Serves to lower strata

Stratum 3:

Reads from stratum 2; Serves to lower strata

Stratum $n+1$:

Reads from stratum n ; Serves to lower strata

Stratum 15:

Reads from stratum 14; This is the lowest stratum.

This process continues down to Stratum 15 which is the lowest valid stratum. The label Stratum 16 is used to indicate an unsynchronized state.

14.3. Understanding NTP

The version of **NTP** used by Red Hat Enterprise Linux is as described in [RFC 1305 Network Time Protocol \(Version 3\) Specification, Implementation and Analysis](#) and [RFC 5905 Network Time Protocol Version 4: Protocol and Algorithms Specification](#)

This implementation of **NTP** enables sub-second accuracy to be achieved. Over the Internet, accuracy to 10s of milliseconds is normal. On a Local Area Network (LAN), 1 ms accuracy is possible under ideal conditions. This is because clock drift is now accounted and corrected for, which was not done in earlier, simpler, time protocol systems. A resolution of 233 picoseconds is provided by using 64-bit timestamps: 32-bits for seconds, 32-bits for fractional seconds.

NTP represents the time as a count of the number of seconds since 00:00 (midnight) 1 January, 1900 GMT. As 32-bits is used to count the seconds, this means the time will “roll over” in 2036. However **NTP** works on the difference between timestamps so this does not present the same level of problem as other implementations of time protocols have done. If a hardware clock accurate to better than 68 years is available at boot time then **NTP** will correctly interpret the current date. The **NTP4** specification provides for an “Era Number” and an “Era Offset” which can be used to make software more robust when dealing with time lengths of more than 68 years. Note, please do not confuse this with the Unix Year 2038 problem.

The **NTP** protocol provides additional information to improve accuracy. Four timestamps are used to allow the calculation of round-trip time and server response time. In order for a system in its role as **NTP** client to synchronize with a reference time server, a packet is sent with an “originate timestamp”. When the packet arrives, the time server adds a “receive timestamp”. After processing the request for time and date information and just before returning the packet, it adds a “transmit timestamp”. When the returning packet arrives at the **NTP** client, a “receive timestamp” is generated. The client can now calculate the total round trip time and by subtracting the processing time derive the actual traveling time. By assuming the outgoing and return trips take equal time, the single-trip delay in receiving the **NTP** data is calculated. The full **NTP** algorithm is much more complex than presented here.

Each packet containing time information received is not immediately acted upon, but is subject to validation checks and then used together with several other samples to arrive at a reasonably good estimate of the time. This is then compared to the system clock to determine the time offset, that is to say, the difference between the system clock's time and what **ntpd** has determined the time should be. The system clock is adjusted slowly, at most at a rate of 0.5ms per second, to reduce this offset by changing the frequency of the counter being used. It will take at least 2000 seconds to adjust the clock by 1 second using this method. This slow change is referred to as slewing and cannot go backwards. If the time offset of the clock is more than 128ms (the default setting), **ntpd** can “step” the clock forwards or backwards. If the time offset at system start is greater than 1000 seconds then the user, or an installation script, should make a manual adjustment. See [Chapter 2, Configuring the Date and Time](#). With the **-g** option to the **ntpd** command (used by default), any offset at system start will be corrected, but during normal operation only offsets of up to 1000 seconds will be corrected.

Some software may fail or produce an error if the time is changed backwards. For systems that are sensitive to step changes in the time, the threshold can be changed to 600s instead of 128ms using the **-x** option (unrelated to the **-g** option). Using the **-x** option to increase the stepping limit from 0.128s to 600s has a drawback because a different method of controlling the clock has to be used. It disables the kernel clock discipline and may have a negative impact on the clock accuracy. The **-x** option can be added to the `/etc/sysconfig/ntp`d configuration file.

14.4. Understanding the Drift File

The drift file is used to store the frequency offset between the system clock running at its nominal frequency and the frequency required to remain in synchronization with UTC. If present, the value contained in the drift file is read at system start and used to correct the clock source. Use of the drift file reduces the time required to achieve a stable and accurate time. The value is calculated, and the drift file replaced, once per hour by **ntpd**. The drift file is replaced, rather than just updated, and for this reason the drift file must be in a directory for which the **ntpd** has write permissions.

14.5. UTC, Timezones, and DST

As **NTP** is entirely in UTC (Universal Time, Coordinated), Timezones and DST (Daylight Saving Time) are applied locally by the system. The file `/etc/localtime` is a copy of, or symlink to, a zone information file from `/usr/share/zoneinfo`. The RTC may be in localtime or in UTC, as specified by the 3rd line of `/etc/adjtime`, which will be one of LOCAL or UTC to indicate how the RTC clock has been set. Users can easily change this setting using the checkbox **System Clock Uses UTC** in the **Date and Time** graphical configuration tool. See [Chapter 2, Configuring the Date and Time](#) for information on how to use that tool. Running the RTC in UTC is recommended to avoid various problems when daylight saving time is changed.

The operation of **ntpd** is explained in more detail in the man page **ntpd(8)**. The resources section lists useful sources of information. See [Section 14.20, “Additional Resources”](#).

14.6. Authentication Options for NTP

NTPv4 added support for the Autokey Security Architecture, which is based on public asymmetric cryptography while retaining support for symmetric key cryptography. The Autokey Security Architecture is described in [RFC 5906 Network Time Protocol Version 4: Autokey Specification](#). The man page **ntp_auth(5)** describes the authentication options and commands for **ntpd**.

An attacker on the network can attempt to disrupt a service by sending **NTP** packets with incorrect time information. On systems using the public pool of **NTP** servers, this risk is mitigated by having more than three **NTP** servers in the list of public **NTP** servers in **/etc/ntp.conf**. If only one time source is compromised or spoofed, **ntpd** will ignore that source. You should conduct a risk assessment and consider the impact of incorrect time on your applications and organization. If you have internal time sources you should consider steps to protect the network over which the **NTP** packets are distributed. If you conduct a risk assessment and conclude that the risk is acceptable, and the impact to your applications minimal, then you can choose not to use authentication.

The broadcast and multicast modes require authentication by default. If you have decided to trust the network then you can disable authentication by using **disable auth** directive in the **ntp.conf** file. Alternatively, authentication needs to be configured by using SHA1 or MD5 symmetric keys, or by public (asymmetric) key cryptography using the Autokey scheme. The Autokey scheme for asymmetric cryptography is explained in the **ntp_auth(8)** man page and the generation of keys is explained in **ntp-keygen(8)**. To implement symmetric key cryptography, see [Section 14.17.12, “Configuring Symmetric Authentication Using a Key”](#) for an explanation of the **key** option.

14.7. Managing the Time on Virtual Machines

Virtual machines cannot access a real hardware clock and a virtual clock is not stable enough as the stability is dependent on the host systems work load. For this reason, para-virtualized clocks should be provided by the virtualization application in use. On Red Hat Enterprise Linux with **KVM** the default clock source is **kvm-clock**. See the [KVM guest timing management](#) chapter of the *Virtualization Host Configuration and Guest Installation Guide*.

14.8. Understanding Leap Seconds

Greenwich Mean Time (GMT) was derived by measuring the solar day, which is dependent on the Earth's rotation. When atomic clocks were first made, the potential for more accurate definitions of time became possible. In 1958, International Atomic Time (TAI) was introduced based on the more accurate and very stable atomic clocks. A more accurate astronomical time, Universal Time 1 (UT1), was also introduced to replace GMT. The atomic clocks are in fact far more stable than the rotation of the Earth and so the two times began to drift apart. For this reason UTC was introduced as a practical measure. It is kept within one second of UT1 but to avoid making many small trivial adjustments it was decided to introduce the concept of a *leap second* in order to reconcile the difference in a manageable way. The difference between UT1 and UTC is monitored until they drift apart by more than half a second. Then only is it deemed necessary to introduce a one second adjustment, forward or backward. Due to the erratic nature of the Earth's rotational speed, the need for an adjustment cannot be predicted far into the future. The decision as to when to make an adjustment is made by the [International Earth Rotation and Reference Systems Service \(IERS\)](#). However, these announcements are important only to administrators of Stratum 1 servers because **NTP** transmits information about pending leap seconds and applies them automatically.

14.9. Understanding the ntpd Configuration File

The daemon, **ntpd**, reads the configuration file at system start or when the service is restarted. The

default location for the file is **/etc/ntp.conf** and you can view the file by entering the following command:

```
~]$ less /etc/ntp.conf
```

The configuration commands are explained briefly later in this chapter, see [Section 14.17, “Configure NTP”](#), and more verbosely in the **ntp.conf(5)** man page.

Here follows a brief explanation of the contents of the default configuration file:

The driftfile entry

A path to the drift file is specified, the default entry on Red Hat Enterprise Linux is:

```
driftfile /var/lib/ntp/drift
```

If you change this be certain that the directory is writable by **ntpd**. The file contains one value used to adjust the system clock frequency after every system or service start. See [Understanding the Drift File](#) for more information.

The access control entries

The following line sets the default access control restriction:

```
restrict default nomodify notrap nopeer noquery
```

The **nomodify** options prevents any changes to the configuration. The **notrap** option prevents **ntpdc** control message protocol traps. The **nopeer** option prevents a peer association being formed. The **noquery** option prevents **ntpq** and **ntpdc** queries, but not time queries, from being answered. The **ntpq** and **ntpdc** queries can be used in amplification attacks (see [CVE-2013-5211](#) for more details), do not remove the **noquery** option from the **restrict default** command on publicly accessible systems.

Addresses within the range **127.0.0.0/8** range are sometimes required by various processes or applications. As the "restrict default" line above prevents access to everything not explicitly allowed, access to the standard loopback address for **IPv4** and **IPv6** is permitted by means of the following lines:

```
# the administrative functions.
restrict 127.0.0.1
restrict ::1
```

Addresses can be added underneath if specifically required by another application.

Hosts on the local network are not permitted because of the "restrict default" line above. To change this, for example to allow hosts from the **192.0.2.0/24** network to query the time and statistics but nothing more, a line in the following format is required:

```
restrict 192.0.2.0 mask 255.255.255.0 nomodify notrap nopeer
```

To allow unrestricted access from a specific host, for example **192.0.2.250/24**, a line in the following format is required:

```
restrict 192.0.2.250
```

A mask of **255.255.255.255** is applied if none is specified.

The restrict commands are explained in the **ntp_acc(5)** man page.

The public servers entry

By default, the **ntp.conf** file contains four public server entries:

```
server 0.rhel.pool.ntp.org iburst
server 1.rhel.pool.ntp.org iburst
server 2.rhel.pool.ntp.org iburst
server 3.rhel.pool.ntp.org iburst
```

The broadcast multicast servers entry

By default, the **ntp.conf** file contains some commented out examples. These are largely self explanatory. See the explanation of the specific commands [Section 14.17, “Configure NTP”](#). If required, add your commands just below the examples.



Note

When the **DHCP** client program, **dhclient**, receives a list of **NTP** servers from the **DHCP** server, it adds them to **ntp.conf** and restarts the service. To disable that feature, add **PEERNTP=no** to **/etc/sysconfig/network**.

14.10. Understanding the ntpd Sysconfig File

The file will be read by the **ntpd** init script on service start. The default contents is as follows:

```
# Command line options for ntpd
OPTIONS="-g"
```

The **-g** option enables **ntpd** to ignore the offset limit of 1000s and attempt to synchronize the time even if the offset is larger than 1000s, but only on system start. Without that option **ntpd** will exit if the time offset is greater than 1000s. It will also exit after system start if the service is restarted and the offset is greater than 1000s even with the **-g** option.

14.11. Disabling chrony

In order to use **ntpd** the default user space daemon, **chronyd**, must be stopped and disabled. Issue the following command as **root**:

```
~]# systemctl stop chronyd
```

To prevent it restarting at system start, issue the following command as **root**:

```
~]# systemctl disable chronyd
```

To check the status of **chronyd**, issue the following command:

```
~]$ systemctl status chronyd
```

14.12. Checking if the NTP Daemon is Installed

To check if **ntpd** is installed, enter the following command as **root**:

```
~]# yum install ntp
```

NTP is implemented by means of the daemon or service **ntpd**, which is contained within the *ntp* package.

14.13. Installing the NTP Daemon (ntpd)

To install **ntpd**, enter the following command as **root**:

```
~]# yum install ntp
```

To enable **ntpd** at system start, enter the following command as **root**:

```
~]# systemctl enable ntpd
```

14.14. Checking the Status of NTP

To check if **ntpd** is running and configured to run at system start, issue the following command:

```
~]$ systemctl status ntpd
```

To obtain a brief status report from **ntpd**, issue the following command:

```
~]$ ntpstat
unsynchronised
time server re-starting
polling server every 64 s
```

```
~]$ ntpstat
synchronised to NTP server (10.5.26.10) at stratum 2
    time correct to within 52 ms
    polling server every 1024 s
```

14.15. Configure the Firewall to Allow Incoming NTP Packets

The **NTP** traffic consists of **UDP** packets on port **123** and needs to be permitted through network and host-based firewalls in order for **NTP** to function.

Check if the firewall is configured to allow incoming **NTP** traffic for clients using the graphical **Firewall Configuration** tool.

To start the graphical **firewall-config** tool, press the **Super** key to enter the Activities Overview, type **firewall** and then press **Enter**. The **firewall-config** tool appears. You will be prompted for your user password.

To start the graphical firewall configuration tool using the command line, enter the following command as **root** user:

```
~]# firewall-config
```

The **Firewall Configuration** window opens. Note, this command can be run as normal user but you will then be prompted for the **root** password from time to time.

Look for the word “Connected” in the lower left corner. This indicates that the **firewall-config** tool is connected to the user space daemon, **firewalld**.

14.15.1. Change the Firewall Settings

To immediately change the current firewall settings, ensure the current view is set to **Runtime Configuration**. Alternatively, to edit the settings to be applied at the next system start, or firewall reload, select **Permanent Configuration** from the drop-down list.



Note

When making changes to the firewall settings in **Runtime Configuration** mode, your selection takes immediate effect when you set or clear the check box associated with the service. You should keep this in mind when working on a system that may be in use by other users.

When making changes to the firewall settings in **Permanent Configuration** mode, your selection will only take effect when you reload the firewall or the system restarts. You can use the reload icon below the **File** menu, or click the **Options** menu and select **Reload Firewall**.

14.15.2. Open Ports in the Firewall for NTP Packets

To permit traffic through the firewall to a certain port, start the **firewall-config** tool and select the network zone whose settings you want to change. Select the **Ports** tab and the click the **Add** button on the right hand side. The **Port and Protocol** window opens.

Enter the port number **123** and select **udp** from the drop-down list.

14.16. Configure ntpdate Servers

The purpose of the **ntpdate** service is to set the clock during system boot. This was used previously to ensure that the services started after **ntpdate** would have the correct time and not observe a jump in the clock. The use of **ntpdate** and the list of step-tickers is considered deprecated and so Red Hat Enterprise Linux 7 uses the **-g** option to the **ntpd** command and not **ntpdate** by default.

The **ntpdate** service in Red Hat Enterprise Linux 7 is mostly useful only when used alone without **ntpd**. With **systemd**, which starts services in parallel, enabling the **ntpdate** service will not ensure that other services started after it will have correct time unless they specify an ordering dependency on **time-sync.target**, which is provided by the **ntpdate** service. In order to ensure a service starts with correct time, add **After=time-sync.target** to the service and enable one of the services which provide the target (**ntpdate** or **sntp**). Some services on Red Hat Enterprise Linux 7 have the dependency included by default (for example, **dhcpd**, **dhcpd6**, and **crond**).

To check if the **ntpdate** service is enabled to run at system start, issue the following command:

```
~]$ systemctl status ntpdate
```

To enable the service to run at system start, issue the following command as **root**:

```
~]# systemctl enable ntpdate
```

In Red Hat Enterprise Linux 7 the default **/etc/ntp/step-tickers** file contains **0.rhel.pool.ntp.org**. To configure additional **ntpdate** servers, using a text editor running as **root**, edit **/etc/ntp/step-tickers**. The number of servers listed is not very important as **ntpdate** will only use this to obtain the date information once when the system is starting. If you have an internal time server then use that host name for the first line. An additional host on the second line as a backup is sensible. The selection of backup servers and whether the second host is internal or external depends on your risk assessment. For example, what is the chance of any problem affecting the first server also affecting the second server? Would connectivity to an external server be more likely to be available than connectivity to internal servers in the event of a network failure disrupting access to the first server?

14.17. Configure NTP

To change the default configuration of the **NTP** service, use a text editor running as **root** user to edit the **/etc/ntp.conf** file. This file is installed together with **ntpd** and is configured to use time servers from the Red Hat pool by default. The man page **ntp.conf(5)** describes the command options that can be used in the configuration file apart from the access and rate limiting commands which are explained in the **ntp_acc(5)** man page.

14.17.1. Configure Access Control to an NTP Service

To restrict or control access to the **NTP** service running on a system, make use of the **restrict** command in the **ntp.conf** file. See the commented out example:

```
# Hosts on local network are less restricted.  
#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
```

The **restrict** command takes the following form:

```
restrict option
```

where *option* is one or more of:

- ▶ **ignore** — All packets will be ignored, including **ntpq** and **ntpdc** queries.
- ▶ **kod** — a “Kiss-o'-death” packet is to be sent to reduce unwanted queries.
- ▶ **limited** — do not respond to time service requests if the packet violates the rate limit default values or those specified by the **discard** command. **ntpq** and **ntpdc** queries are not affected. For more information on the **discard** command and the default values, see [Section 14.17.2, “Configure Rate Limiting Access to an NTP Service”](#).
- ▶ **lowpriotrap** — traps set by matching hosts to be low priority.
- ▶ **nomodify** — prevents any changes to the configuration.
- ▶ **noquery** — prevents **ntpq** and **ntpdc** queries, but not time queries, from being answered.
- ▶ **nopeer** — prevents a peer association being formed.
- ▶ **noserve** — deny all packets except **ntpq** and **ntpdc** queries.

- ▶ **notrap** — prevents **ntpdc** control message protocol traps.
- ▶ **notrust** — deny packets that are not cryptographically authenticated.
- ▶ **ntpport** — modify the match algorithm to only apply the restriction if the source port is the standard **NTP UDP** port **123**.
- ▶ **version** — deny packets that do not match the current **NTP** version.

To configure rate limit access to not respond at all to a query, the respective **restrict** command has to have the **limited** option. If **ntp**d should reply with a **KoD** packet, the **restrict** command needs to have both **limited** and **kod** options.

The **ntpq** and **ntpdc** queries can be used in amplification attacks (see [CVE-2013-5211](#) for more details), do not remove the **noquery** option from the **restrict default** command on publicly accessible systems.

14.17.2. Configure Rate Limiting Access to an NTP Service

To enable rate limiting access to the **NTP** service running on a system, add the **limited** option to the **restrict** command as explained in [Section 14.17.1, “Configure Access Control to an NTP Service”](#). If you do not want to use the default discard parameters, then also use the **discard** command as explained here.

The **discard** command takes the following form:

```
discard [average value] [minimum value] [monitor value]
```

- ▶ **average** — specifies the minimum average packet spacing to be permitted, it accepts an argument in \log_2 seconds. The default value is 3 (2^3 equates to 8 seconds).
- ▶ **minimum** — specifies the minimum packet spacing to be permitted, it accepts an argument in \log_2 seconds. The default value is 1 (2^1 equates to 2 seconds).
- ▶ **monitor** — specifies the discard probability for packets once the permitted rate limits have been exceeded. The default value is 3000 seconds. This option is intended for servers that receive 1000 or more requests per second.

Examples of the **discard** command are as follows:

```
discard average 4
```

```
discard average 4 minimum 2
```

14.17.3. Adding a Peer Address

To add the address of a peer, that is to say, the address of a server running an **NTP** service of the same stratum, make use of the **peer** command in the **ntp.conf** file.

The **peer** command takes the following form:

```
peer address
```

where **address** is an **IP** unicast address or a **DNS** resolvable name. The address must only be that of a system known to be a member of the same stratum. Peers should have at least one time source that is different to each other. Peers are normally systems under the same administrative control.

14.17.4. Adding a Server Address

To add the address of a server, that is to say, the address of a server running an **NTP** service of a higher stratum, make use of the **server** command in the **ntp.conf** file.

The **server** command takes the following form:

```
server address
```

where **address** is an **IP** unicast address or a **DNS** resolvable name. The address of a remote reference server or local reference clock from which packets are to be received.

14.17.5. Adding a Broadcast or Multicast Server Address

To add a broadcast or multicast address for sending, that is to say, the address to broadcast or multicast **NTP** packets to, make use of the **broadcast** command in the **ntp.conf** file.

The broadcast and multicast modes require authentication by default. See [Section 14.6, “Authentication Options for NTP”](#).

The **broadcast** command takes the following form:

```
broadcast address
```

where **address** is an **IP** broadcast or multicast address to which packets are sent.

This command configures a system to act as an **NTP** broadcast server. The address used must be a broadcast or a multicast address. Broadcast address implies the **IPv4** address **255.255.255.255**. By default, routers do not pass broadcast messages. The multicast address can be an **IPv4** Class D address, or an **IPv6** address. The IANA has assigned **IPv4** multicast address **224.0.1.1** and **IPv6** address **FF00::101** (site local) to **NTP**. Administratively scoped**IPv4** multicast addresses can also be used, as described in [RFC 2365 Administratively Scoped IP Multicast](#).

14.17.6. Adding a Manycast Client Address

To add a manycast client address, that is to say, to configure a multicast address to be used for **NTP** server discovery, make use of the **manycastclient** command in the **ntp.conf** file.

The **manycastclient** command takes the following form:

```
manycastclient address
```

where **address** is an **IP** multicast address from which packets are to be received. The client will send a request to the address and select the best servers from the responses and ignore other servers. **NTP** communication then uses unicast associations, as if the discovered **NTP** servers were listed in **ntp.conf**.

This command configures a system to act as an **NTP** client. Systems can be both client and server at the same time.

14.17.7. Adding a Broadcast Client Address

To add a broadcast client address, that is to say, to configure a broadcast address to be monitored for broadcast **NTP** packets, make use of the **broadcastclient** command in the **ntp.conf** file.

The **broadcastclient** command takes the following form:

```
broadcastclient
```

Enables the receiving of broadcast messages. Requires authentication by default. See [Section 14.6, “Authentication Options for NTP”](#).

This command configures a system to act as an **NTP** client. Systems can be both client and server at the same time.

14.17.8. Adding a Manycast Server Address

To add a manycast server address, that is to say, to configure an address to allow the clients to discover the server by multicasting **NTP** packets, make use of the **manycastserver** command in the **ntp.conf** file.

The **manycastserver** command takes the following form:

```
manycastserver address
```

Enables the sending of multicast messages. Where *address* is the address to multicast to. This should be used together with authentication to prevent service disruption.

This command configures a system to act as an **NTP** server. Systems can be both client and server at the same time.

14.17.9. Adding a Multicast Client Address

To add a multicast client address, that is to say, to configure a multicast address to be monitored for multicast **NTP** packets, make use of the **multicastclient** command in the **ntp.conf** file.

The **multicastclient** command takes the following form:

```
multicastclient address
```

Enables the receiving of multicast messages. Where *address* is the address to subscribe to. This should be used together with authentication to prevent service disruption.

This command configures a system to act as an **NTP** client. Systems can be both client and server at the same time.

14.17.10. Configuring the Burst Option

Using the **burst** option against a public server is considered abuse. Do not use this option with public **NTP** servers. Use it only for applications within your own organization.

To increase the average quality of time offset statistics, add the following option to the end of a server command:

```
burst
```

At every poll interval, send a burst of eight packets instead of one, when the server is responding. For use with the **server** command to improve the average quality of the time offset calculations.

14.17.11. Configuring the **iburst** Option

To improve the time taken for initial synchronization, add the following option to the end of a server command:

iburst

At every poll interval, send a burst of eight packets instead of one. When the server is not responding, packets are sent 16s apart. When the server responds, packets are sent every 2s. For use with the **server** command to improve the time taken for initial synchronization. This is now a default option in the configuration file.

14.17.12. Configuring Symmetric Authentication Using a Key

To configure symmetric authentication using a key, add the following option to the end of a server or peer command:

key *number*

where *number* is in the range **1** to **65534** inclusive. This option enables the use of a *message authentication code* (MAC) in packets. This option is for use with the **peer**, **server**, **broadcast**, and **multicastclient** commands.

The option can be used in the **/etc/ntp.conf** file as follows:

```
server 192.168.1.1 key 10
broadcast 192.168.1.255 key 20
multicastclient 239.255.254.254 key 30
```

See also [Section 14.6, “Authentication Options for NTP”](#).

14.17.13. Configuring the Poll Interval

To change the default poll interval, add the following options to the end of a server or peer command:

minpoll *value* and **maxpoll** *value*

Options to change the default poll interval, where the interval in seconds will be calculated by raising 2 to the power of *value*, in other words, the interval is expressed in \log_2 seconds. The default **minpoll** value is 6, 2^6 equates to 64s. The default value for **maxpoll** is 10, which equates to 1024s. Allowed values are in the range 3 to 17 inclusive, which equates to 8s to 36.4h respectively. These options are for use with the **peer** or **server**. Setting a shorter **maxpoll** may improve clock accuracy.

14.17.14. Configuring Server Preference

To specify that a particular server should be preferred above others of similar statistical quality, add the following option to the end of a server or peer command:

prefer

Use this server for synchronization in preference to other servers of similar statistical quality. This option is for use with the **peer** or **server** commands.

14.17.15. Configuring the Time-to-Live for NTP Packets

To specify that a particular *time-to-live* (TTL) value should be used in place of the default, add the following option to the end of a server or peer command:

```
ttl value
```

Specify the time-to-live value to be used in packets sent by broadcast servers and multicast **NTP** servers. Specify the maximum time-to-live value to use for the “expanding ring search” by a manycast client. The default value is **127**.

14.17.16. Configuring the NTP Version to Use

To specify that a particular version of **NTP** should be used in place of the default, add the following option to the end of a server or peer command:

```
version value
```

Specify the version of **NTP** set in created **NTP** packets. The value can be in the range **1** to **4**. The default is **4**.

14.18. Configuring the Hardware Clock Update

To configure the system clock to update the hardware clock, also known as the real-time clock (RTC), once after executing **ntpdate**, add the following line to **/etc/sysconfig/ntpdate**:

```
SYNC_HWCLOCK=yes
```

To update the hardware clock from the system clock, issue the following command as **root**:

```
~]# hwclock --systohc
```

When the system clock is being synchronized by **ntpd**, the kernel will in turn update the RTC every 11 minutes automatically.

14.19. Configuring Clock Sources

To list the available clock sources on your system, issue the following commands:

```
~]$ cd /sys/devices/system/clocksource/clocksource0/
clocksource0]$ cat available_clocksource
kvm-clock tsc hpet acpi_pm
clocksource0]$ cat current_clocksource
kvm-clock
```

In the above example, the kernel is using **kvm-clock**. This was selected at boot time as this is a virtual machine.

To override the default clock source, add a line similar to the following in **grub.conf**:

```
clocksource=tsc
```

The available clock source is architecture dependent.

14.20. Additional Resources

The following sources of information provide additional resources regarding **NTP** and **ntpd**.

14.20.1. Installed Documentation

- ▶ **ntpd(8)** man page — Describes **ntpd** in detail, including the command line options.
- ▶ **ntp.conf(5)** man page — Contains information on how to configure associations with servers and peers.
- ▶ **ntpq(8)** man page — Describes the **NTP** query utility for monitoring and querying an **NTP** server.
- ▶ **ntpdc(8)** man page — Describes the **ntpdc** utility for querying and changing the state of **ntpd**.
- ▶ **ntp_auth(5)** man page — Describes authentication options, commands, and key management for **ntpd**.
- ▶ **ntp_keygen(8)** man page — Describes generating public and private keys for **ntpd**.
- ▶ **ntp_acc(5)** man page — Describes access control options using the **restrict** command.
- ▶ **ntp_mon(5)** man page — Describes monitoring options for the gathering of statistics.
- ▶ **ntp_clock(5)** man page — Describes commands for configuring reference clocks.
- ▶ **ntp_misc(5)** man page — Describes miscellaneous options.
- ▶ **ntp_decode(5)** man page — Lists the status words, event messages and error codes used for **ntpd** reporting and monitoring.
- ▶ **ntpstat(8)** man page — Describes a utility for reporting the synchronization state of the **NTP** daemon running on the local machine.
- ▶ **ntptime(8)** man page — Describes a utility for reading and setting kernel time variables.
- ▶ **tickadj(8)** man page — Describes a utility for reading, and optionally setting, the length of the tick.

14.20.2. Useful Websites

<http://doc.ntp.org/>

The NTP Documentation Archive

<http://www.eecis.udel.edu/~mills/ntp.html>

Network Time Synchronization Research Project.

<http://www.eecis.udel.edu/~mills/ntp/html/manopt.html>

Information on Automatic Server Discovery in **NTPv4**.

Chapter 15. Configuring PTP Using `ptp4l`

15.1. Introduction to PTP

The *Precision Time Protocol* (PTP) is a protocol used to synchronize clocks in a network. When used in conjunction with hardware support, **PTP** is capable of sub-microsecond accuracy, which is far better than is normally obtainable with **NTP**. **PTP** support is divided between the kernel and user space. The kernel in Red Hat Enterprise Linux includes support for **PTP** clocks, which are provided by network drivers. The actual implementation of the protocol is known as **linuxptp**, a **PTPv2** implementation according to the IEEE standard 1588 for Linux.

The *linuxptp* package includes the **ptp4l** and **phc2sys** programs for clock synchronization. The **ptp4l** program implements the **PTP** boundary clock and ordinary clock. With hardware time stamping, it is used to synchronize the **PTP** hardware clock to the master clock, and with software time stamping it synchronizes the system clock to the master clock. The **phc2sys** program is needed only with hardware time stamping, for synchronizing the system clock to the **PTP** hardware clock on the *network interface card* (NIC).

15.1.1. Understanding PTP

The clocks synchronized by **PTP** are organized in a master-slave hierarchy. The slaves are synchronized to their masters which may be slaves to their own masters. The hierarchy is created and updated automatically by the *best master clock* (BMC) algorithm, which runs on every clock. When a clock has only one port, it can be *master* or *slave*, such a clock is called an *ordinary clock* (OC). A clock with multiple ports can be master on one port and slave on another, such a clock is called a *boundary clock* (BC). The top-level master is called the *grandmaster clock*, which can be synchronized by using a *Global Positioning System* (GPS) time source. By using a GPS-based time source, disparate networks can be synchronized with a high-degree of accuracy.

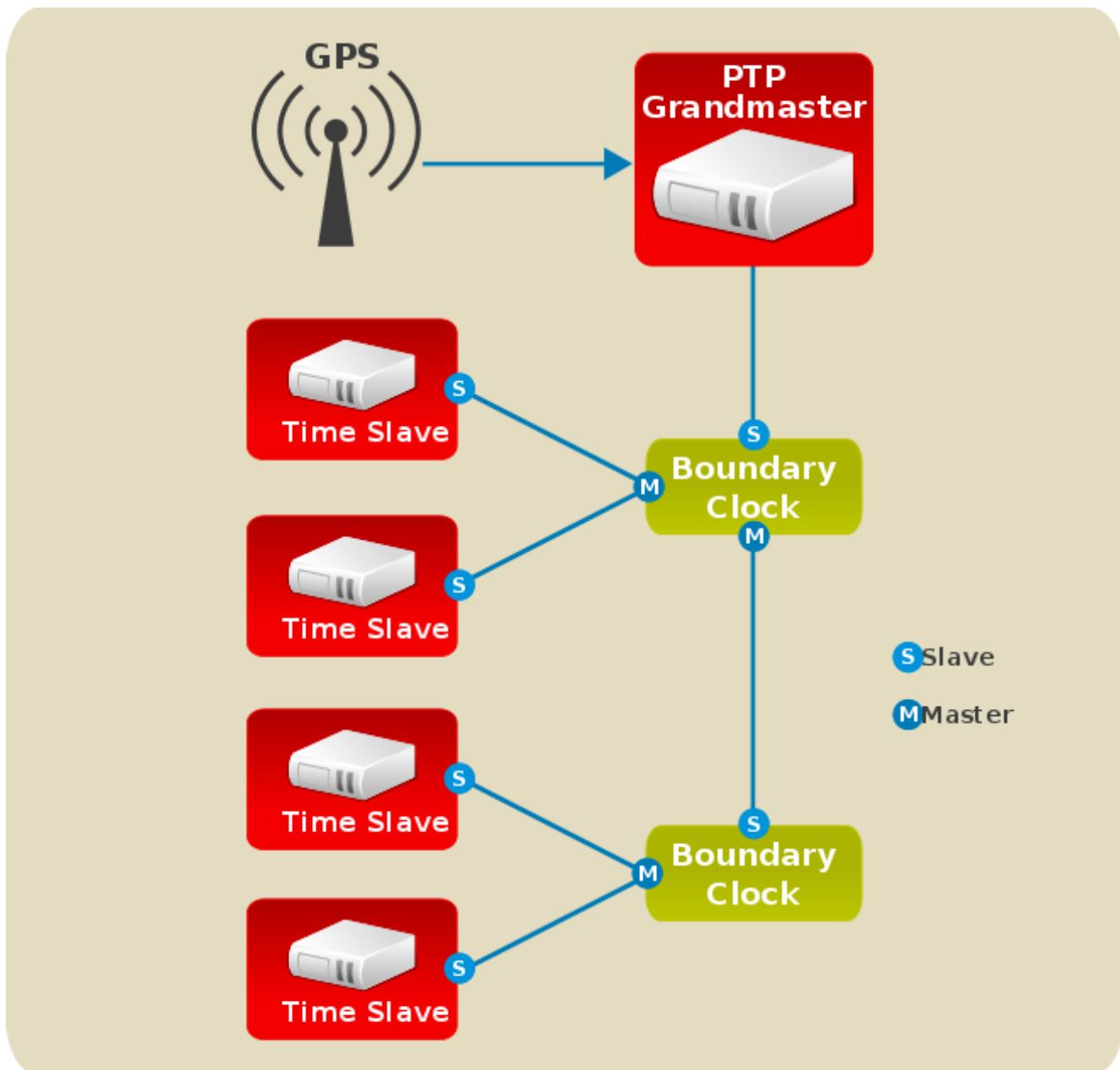


Figure 15.1. PTP grandmaster, boundary, and slave Clocks

15.1.2. Advantages of PTP

One of the main advantages that **PTP** has over the *Network Time Protocol* (NTP) is hardware support present in various *network interface controllers* (NIC) and network switches. This specialized hardware allows **PTP** to account for delays in message transfer, and greatly improves the accuracy of time synchronization. While it is possible to use non-PTP enabled hardware components within the network, this will often cause an increase in jitter or introduce an asymmetry in the delay resulting in synchronization inaccuracies, which add up with multiple non-PTP aware components used in the communication path. To achieve the best possible accuracy, it is recommended that all networking components between **PTP** clocks are **PTP** hardware enabled. Time synchronization in larger networks where not all of the networking hardware supports **PTP** might be better suited for **NTP**.

With hardware **PTP** support, the NIC has its own on-board clock, which is used to time stamp the received and transmitted **PTP** messages. It is this on-board clock that is synchronized to the **PTP** master, and the computer's system clock is synchronized to the **PTP** hardware clock on the NIC. With software **PTP** support, the system clock is used to time stamp the **PTP** messages and it is synchronized to the **PTP**

master directly. Hardware **PTP** support provides better accuracy since the NIC can time stamp the **PTP** packets at the exact moment they are sent and received while software **PTP** support requires additional processing of the **PTP** packets by the operating system.

15.2. Using PTP

In order to use **PTP**, the kernel network driver for the intended interface has to support either software or hardware time stamping capabilities.

15.2.1. Checking for Driver and Hardware Support

In addition to hardware time stamping support being present in the driver, the NIC must also be capable of supporting this functionality in the physical hardware. The best way to verify the time stamping capabilities of a particular driver and NIC is to use the **ethtool** utility to query the interface as follows:

```
~]# ethtool -T eth3
Time stamping parameters for eth3:
Capabilities:
    hardware-transmit      (SOF_TIMESTAMPING_TX_HARDWARE)
    software-transmit      (SOF_TIMESTAMPING_TX_SOFTWARE)
    hardware-receive       (SOF_TIMESTAMPING_RX_HARDWARE)
    software-receive       (SOF_TIMESTAMPING_RX_SOFTWARE)
    software-system-clock  (SOF_TIMESTAMPING_SOFTWARE)
    hardware-raw-clock     (SOF_TIMESTAMPING_RAW_HARDWARE)
PTP Hardware Clock: 0
Hardware Transmit Timestamp Modes:
    off                  (HWTSTAMP_TX_OFF)
    on                   (HWTSTAMP_TX_ON)
Hardware Receive Filter Modes:
    none                (HWTSTAMP_FILTER_NONE)
    all                 (HWTSTAMP_FILTER_ALL)
```

Where *eth3* is the interface you wish to check.

For software time stamping support, the parameters list should include:

- ▶ **SOF_TIMESTAMPING_SOFTWARE**
- ▶ **SOF_TIMESTAMPING_TX_SOFTWARE**
- ▶ **SOF_TIMESTAMPING_RX_SOFTWARE**

For hardware time stamping support, the parameters list should include:

- ▶ **SOF_TIMESTAMPING_RAW_HARDWARE**
- ▶ **SOF_TIMESTAMPING_TX_HARDWARE**
- ▶ **SOF_TIMESTAMPING_RX_HARDWARE**

15.2.2. Installing PTP

The kernel in Red Hat Enterprise Linux includes support for **PTP**. User space support is provided by the tools in the **linuxptp** package. To install **linuxptp**, issue the following command as **root**:

```
~]# yum install linuxptp
```

This will install **ptp4l** and **phc2sys**.

Do not run more than one service to set the system clock's time at the same time. If you intend to serve PTP time using NTP, see [Section 15.7, “Serving PTP Time with NTP”](#).

15.2.3. Starting ptpt41

The **ptpt41** program tries to use hardware time stamping by default. To use **ptpt41** with hardware time stamping capable drivers and NICs, you must provide the network interface to use with the **-i** option. Enter the following command as **root**:

```
~]# ptpt41 -i eth3 -m
```

Where **eth3** is the interface you wish to configure. Below is example output from **ptpt41** when the PTP clock on the NIC is synchronized to a master:

```
~]# ptpt41 -i eth3 -m
selected eth3 as PTP clock
port 1: INITIALIZING to LISTENING on INITIALIZE
port 0: INITIALIZING to LISTENING on INITIALIZE
port 1: new foreign master 00a069.ffffe.0b552d-1
selected best master clock 00a069.ffffe.0b552d
port 1: LISTENING to UNCALIBRATED on RS_SLAVE
master offset -23947 s0 freq +0 path delay      11350
master offset -28867 s0 freq +0 path delay      11236
master offset -32801 s0 freq +0 path delay      10841
master offset -37203 s1 freq +0 path delay      10583
master offset -7275 s2 freq -30575 path delay    10583
port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
master offset -4552 s2 freq -30035 path delay    10385
```

The master offset value is the measured offset from the master in nanoseconds. The **s0**, **s1**, **s2** strings indicate the different clock servo states: **s0** is unlocked, **s1** is clock step and **s2** is locked. Once the servo is in the locked state (**s2**), the clock will not be stepped (only slowly adjusted) unless the **pi_offset_const** option is set to a positive value in the configuration file (described in the **ptpt41(8)** man page). The **adj** value is the frequency adjustment of the clock in parts per billion (ppb). The path delay value is the estimated delay of the synchronization messages sent from the master in nanoseconds. Port 0 is a Unix domain socket used for local PTP management. Port 1 is the **eth3** interface (based on the example above.) INITIALIZING, LISTENING, UNCALIBRATED and SLAVE are some of possible port states which change on the INITIALIZE, RS_SLAVE, MASTER_CLOCK_SELECTED events. In the last state change message, the port state changed from UNCALIBRATED to SLAVE indicating successful synchronization with a PTP master clock.

The **ptpt41** program can also be started as a service by running:

```
~]# systemctl start ptpt41
```

When running as a service, options are specified in the **/etc/sysconfig/ptpt41** file. More information on the different **ptpt41** options and the configuration file settings can be found in the **ptpt41(8)** man page.

By default, messages are sent to **/var/log/messages**. However, specifying the **-m** option enables logging to standard output which can be useful for debugging purposes.

To enable software time stamping, the **-S** option needs to be used as follows:

```
~]# ptpt41 -i eth3 -m -S
```

15.2.3.1. Selecting a Delay Measurement Mechanism

There are two different delay measurement mechanisms and they can be selected by means of an option added to the **ptp4l** command as follows:

-P

The **-P** selects the *peer-to-peer* (P2P) delay measurement mechanism.

The P2P mechanism is preferred as it reacts to changes in the network topology faster, and may be more accurate in measuring the delay, than other mechanisms. The P2P mechanism can only be used in topologies where each port exchanges PTP messages with at most one other P2P port. It must be supported and used by all hardware, including transparent clocks, on the communication path.

-E

The **-E** selects the *end-to-end* (E2E) delay measurement mechanism. This is the default.

The E2E mechanism is also referred to as the delay “request-response” mechanism.

-A

The **-A** enables automatic selection of the delay measurement mechanism.

The automatic option starts **ptp4l** in E2E mode. It will change to P2P mode if a peer delay request is received.



Note

All clocks on a single **PTP** communication path must use the same mechanism to measure the delay. A warning will be printed when a peer delay request is received on a port using the E2E mechanism. A warning will be printed when a E2E delay request is received on a port using the P2P mechanism.

15.3. Specifying a Configuration File

The command line options and other options, which cannot be set on the command line, can be set in an optional configuration file.

No configuration file is read by default, so it needs to be specified at runtime with the **-f** option. For example:

```
~]# ptp4l -f /etc/ptp4l.conf
```

A configuration file equivalent to the **-i eth3 -m -S** options shown above would look as follows:

```
~]# cat /etc/ptp4l.conf
[global]
verbose          1
time_stamping   software
[eth3]
```

15.4. Using the PTP Management Client

The **PTP** management client, **pmc**, can be used to obtain additional information from **ptpt4l** as follows:

```
~]# pmc -u -b 0 'GET CURRENT_DATA_SET'
sending: GET CURRENT_DATA_SET
    90e2ba.ffffe.20c7f8-0 seq 0 RESPONSE MANAGMENT CURRENT_DATA_SET
        stepsRemoved          1
        offsetFromMaster     -142.0
        meanPathDelay       9310.0
```

```
~]# pmc -u -b 0 'GET TIME_STATUS_NP'
sending: GET TIME_STATUS_NP
    90e2ba.ffffe.20c7f8-0 seq 0 RESPONSE MANAGMENT TIME_STATUS_NP
        master_offset          310
        ingress_time           1361545089345029441
        cumulativeScaledRateOffset +1.0000000000
        scaledLastGmPhaseChange 0
        gmTimeBaseIndicator     0
        lastGmPhaseChange      0x0000'0000000000000000.0000
        gmPresent                true
        gmIdentity              00a069.ffffe.0b552d
```

Setting the **-b** option to **zero** limits the boundary to the locally running **ptpt4l** instance. A larger boundary value will retrieve the information also from **PTP** nodes further from the local clock. The retrievable information includes:

- ▶ **stepsRemoved** is the number of communication paths to the grandmaster clock.
- ▶ **offsetFromMaster** and **master_offset** is the last measured offset of the clock from the master in nanoseconds.
- ▶ **meanPathDelay** is the estimated delay of the synchronization messages sent from the master in nanoseconds.
- ▶ if **gmPresent** is true, the **PTP** clock is synchronized to a master, the local clock is not the grandmaster clock.
- ▶ **gmIdentity** is the grandmaster's identity.

For a full list of **pmc** commands, type the following as **root**:

```
~]# pmc help
```

Additional information is available in the **pmc(8)** man page.

15.5. Synchronizing the Clocks

The **phc2sys** program is used to synchronize the system clock to the **PTP** hardware clock (PHC) on the NIC. To start **phc2sys**, where **eth3** is the interface with the **PTP** hardware clock, enter the following command as **root**:

```
~]# phc2sys -s eth3 -w
```

The **-w** option waits for the running **ptpt4l** application to synchronize the **PTP** clock and then retrieves the TAI to UTC offset from **ptpt4l**.

Normally, **PTP** operates in the *International Atomic Time* (TAI) timescale, while the system clock is kept in *Coordinated Universal Time* (UTC). The current offset between the TAI and UTC timescales is 35 seconds. The offset changes when leap seconds are inserted or deleted, which typically happens every few years. The **-O** option needs to be used to set this offset manually when the **-w** is not used, as follows:

```
~]# phc2sys -s eth3 -O -35
```

Once the **phc2sys** servo is in a locked state, the clock will not be stepped, unless the **-S** option is used. This means that the **phc2sys** program should be started after the **ptp4l** program has synchronized the **PTP** hardware clock. However, with **-w**, it is not necessary to start **phc2sys** after **ptp4l** as it will wait for it to synchronize the clock.

The **phc2sys** program can also be started as a service by running:

```
~]# systemctl start phc2sys
```

When running as a service, options are specified in the **/etc/sysconfig/phc2sys** file. More information on the different **phc2sys** options can be found in the **phc2sys(8)** man page.

Note that the examples in this section assume the command is run on a slave system or slave port.

15.6. Verifying Time Synchronization

When **PTP** time synchronization is working properly, new messages with offsets and frequency adjustments will be printed periodically to the **ptp4l** and **phc2sys** (if hardware time stamping is used) outputs. These values will eventually converge after a short period of time. These messages can be seen in **/var/log/messages** file. An example of the output follows:

```
ptp4l[352.359]: selected /dev/ptp0 as PTP clock
ptp4l[352.361]: port 1: INITIALIZING to LISTENING on INITIALIZE
ptp4l[352.361]: port 0: INITIALIZING to LISTENING on INITIALIZE
ptp4l[353.210]: port 1: new foreign master 00a069.ffff.0b552d-1
ptp4l[357.214]: selected best master clock 00a069.ffff.0b552d
ptp4l[357.214]: port 1: LISTENING to UNCALIBRATED on RS_SLAVE
ptp4l[359.224]: master offset      3304 s0 freq      +0 path delay   9202
ptp4l[360.224]: master offset      3708 s1 freq    -29492 path delay   9202
ptp4l[361.224]: master offset     -3145 s2 freq    -32637 path delay   9202
ptp4l[361.224]: port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l[362.223]: master offset     -145 s2 freq    -30580 path delay   9202
ptp4l[363.223]: master offset     1043 s2 freq    -29436 path delay   8972
ptp4l[364.223]: master offset     266 s2 freq    -29900 path delay   9153
ptp4l[365.223]: master offset     430 s2 freq    -29656 path delay   9153
ptp4l[366.223]: master offset     615 s2 freq    -29342 path delay   9169
ptp4l[367.222]: master offset    -191 s2 freq    -29964 path delay   9169
ptp4l[368.223]: master offset     466 s2 freq    -29364 path delay   9170
ptp4l[369.235]: master offset      24 s2 freq    -29666 path delay   9196
ptp4l[370.235]: master offset    -375 s2 freq    -30058 path delay   9238
ptp4l[371.235]: master offset     285 s2 freq    -29511 path delay   9199
ptp4l[372.235]: master offset     -78 s2 freq    -29788 path delay   9204
```

An example of the **phc2sys** output follows:

```
phc2sys[526.527]: Waiting for ptp4l...
phc2sys[527.528]: Waiting for ptp4l...
phc2sys[528.528]: phc offset      55341 s0 freq      +0 delay   2729
phc2sys[529.528]: phc offset      54658 s1 freq    -37690 delay   2725
phc2sys[530.528]: phc offset      888 s2 freq    -36802 delay   2756
```

```

phc2sys[531.528]: phc offset      1156 s2 freq  -36268 delay  2766
phc2sys[532.528]: phc offset      411 s2 freq  -36666 delay  2738
phc2sys[533.528]: phc offset     -73 s2 freq  -37026 delay  2764
phc2sys[534.528]: phc offset      39 s2 freq  -36936 delay  2746
phc2sys[535.529]: phc offset      95 s2 freq  -36869 delay  2733
phc2sys[536.529]: phc offset     -359 s2 freq  -37294 delay  2738
phc2sys[537.529]: phc offset     -257 s2 freq  -37300 delay  2753
phc2sys[538.529]: phc offset     119 s2 freq  -37001 delay  2745
phc2sys[539.529]: phc offset     288 s2 freq  -36796 delay  2766
phc2sys[540.529]: phc offset     -149 s2 freq  -37147 delay  2760
phc2sys[541.529]: phc offset     -352 s2 freq  -37395 delay  2771
phc2sys[542.529]: phc offset     166 s2 freq  -36982 delay  2748
phc2sys[543.529]: phc offset      50 s2 freq  -37048 delay  2756
phc2sys[544.530]: phc offset     -31 s2 freq  -37114 delay  2748
phc2sys[545.530]: phc offset     -333 s2 freq  -37426 delay  2747
phc2sys[546.530]: phc offset     194 s2 freq  -36999 delay  2749

```

For **ptp41** there is also a directive, **summary_interval**, to reduce the output and print only statistics, as normally it will print a message every second or so. For example, to reduce the output to every **1024** seconds, add the following line to the **/etc/ptp41.conf** file:

```
summary_interval 10
```

An example of the **ptp41** output, with **summary_interval 6**, follows:

```

ptp41: [615.253] selected /dev/ptp0 as PTP clock
ptp41: [615.255] port 1: INITIALIZING to LISTENING on INITIALIZE
ptp41: [615.255] port 0: INITIALIZING to LISTENING on INITIALIZE
ptp41: [615.564] port 1: new foreign master 00a069.ffff.0b552d-1
ptp41: [619.574] selected best master clock 00a069.ffff.0b552d
ptp41: [619.574] port 1: LISTENING to UNCALIBRATED on RS_SLAVE
ptp41: [623.573] port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp41: [684.649] rms 669 max 3691 freq -29383 ± 3735 delay 9232 ± 122
ptp41: [748.724] rms 253 max 588 freq -29787 ± 221 delay 9219 ± 158
ptp41: [812.793] rms 287 max 673 freq -29802 ± 248 delay 9211 ± 183
ptp41: [876.853] rms 226 max 534 freq -29795 ± 197 delay 9221 ± 138
ptp41: [940.925] rms 250 max 562 freq -29801 ± 218 delay 9199 ± 148
ptp41: [1004.988] rms 226 max 525 freq -29802 ± 196 delay 9228 ± 143
ptp41: [1069.065] rms 300 max 646 freq -29802 ± 259 delay 9214 ± 176
ptp41: [1133.125] rms 226 max 505 freq -29792 ± 197 delay 9225 ± 159
ptp41: [1197.185] rms 244 max 688 freq -29790 ± 211 delay 9201 ± 162

```

To reduce the output from the **phc2sys**, it can be called it with the **-u** option as follows:

```
~]# phc2sys -u summary-updates
```

Where **summary-updates** is the number of clock updates to include in summary statistics. An example follows:

```

~]# phc2sys -s eth3 -w -m -u 60
phc2sys[700.948]: rms 1837 max 10123 freq -36474 ± 4752 delay 2752 ± 16
phc2sys[760.954]: rms 194 max 457 freq -37084 ± 174 delay 2753 ± 12
phc2sys[820.963]: rms 211 max 487 freq -37085 ± 185 delay 2750 ± 19
phc2sys[880.968]: rms 183 max 440 freq -37102 ± 164 delay 2734 ± 91
phc2sys[940.973]: rms 244 max 584 freq -37095 ± 216 delay 2748 ± 16
phc2sys[1000.979]: rms 220 max 573 freq -36666 ± 182 delay 2747 ± 43
phc2sys[1060.984]: rms 266 max 675 freq -36759 ± 234 delay 2753 ± 17

```

15.7. Serving PTP Time with NTP

The **ntpd** daemon can be configured to distribute the time from the system clock synchronized by **ptp4l** or **phc2sys** by using the LOCAL reference clock driver. To prevent **ntpd** from adjusting the system clock, the **ntp.conf** file must not specify any **NTP** servers. The following is a minimal example of **ntp.conf**:

```
~]# cat /etc/ntp.conf
server 127.127.1.0
fudge 127.127.1.0 stratum 0
```

Note

When the **DHCP** client program, **dhclient**, receives a list of **NTP** servers from the **DHCP** server, it adds them to **ntp.conf** and restarts the service. To disable that feature, add **PEERNTP=no** to **/etc/sysconfig/network**.

15.8. Serving NTP Time with PTP

NTP to **PTP** synchronization in the opposite direction is also possible. When **ntpd** is used to synchronize the system clock, **ptp4l** can be configured with the **priority1** option (or other clock options included in the best master clock algorithm) to be the grandmaster clock and distribute the time from the system clock via **PTP**:

```
~]# cat /etc/ptp4l.conf
[global]
priority1 127
[eth3]
# ptp4l -f /etc/ptp4l.conf
```

With hardware time stamping, **phc2sys** needs to be used to synchronize the **PTP** hardware clock to the system clock:

```
~]# phc2sys -c eth3 -s CLOCK_REALTIME -w
```

To prevent quick changes in the **PTP** clock's frequency, the synchronization to the system clock can be loosened by using smaller **P** (proportional) and **I** (integral) constants of the PI servo:

```
~]# phc2sys -c eth3 -s CLOCK_REALTIME -w -P 0.01 -I 0.0001
```

15.9. Improving Accuracy

Test results indicate that disabling the tickless kernel capability can significantly improve the stability of the system clock, and thus improve the **PTP** synchronization accuracy (at the cost of increased power consumption). The kernel tickless mode can be disabled by adding **nohz=off** to the kernel boot option parameters.

15.10. Additional Resources

The following sources of information provide additional resources regarding **PTP** and the **ptp4l** tools.

15.10.1. Installed Documentation

- » **`ptp4l(8)`** man page — Describes `ptp4l` options including the format of the configuration file.
- » **`pmc(8)`** man page — Describes the **PTP** management client and its command options.
- » **`phc2sys(8)`** man page — Describes a tool for synchronizing the system clock to a **PTP** hardware clock (PHC).

15.10.2. Useful Websites

<http://www.nist.gov/el/isd/ieee/ieee1588.cfm>

The IEEE 1588 Standard.

Part V. Monitoring and Automation

This part describes various tools that allow system administrators to monitor system performance, automate system tasks, and report bugs.

Chapter 16. System Monitoring Tools

In order to configure the system, system administrators often need to determine the amount of free memory, how much free disk space is available, how the hard drive is partitioned, or what processes are running.

16.1. Viewing System Processes

16.1.1. Using the ps Command

The **ps** command allows you to display information about running processes. It produces a static list, that is, a snapshot of what is running when you execute the command. If you want a constantly updated list of running processes, use the **top** command or the **System Monitor** application instead.

To list all processes that are currently running on the system including processes owned by other users, type the following at a shell prompt:

```
ps ax
```

For each listed process, the **ps ax** command displays the process ID (**PID**), the terminal that is associated with it (**TTY**), the current status (**STAT**), the cumulated CPU time (**TIME**), and the name of the executable file (**COMMAND**). For example:

```
~]$ ps ax
 PID TTY      STAT   TIME COMMAND
  1 ?        Ss     0:01 /sbin/init
  2 ?        S      0:00 [kthreadd]
  3 ?        S      0:00 [migration/0]
  4 ?        S      0:00 [ksftirqd/0]
  5 ?        S      0:00 [migration/0]
  6 ?        S      0:00 [watchdog/0]
[output truncated]
```

To display the owner alongside each process, use the following command:

```
ps aux
```

Apart from the information provided by the **ps ax** command, **ps aux** displays the effective username of the process owner (**USER**), the percentage of the CPU (**%CPU**) and memory (**%MEM**) usage, the virtual memory size in kilobytes (**VSZ**), the non-swapped physical memory size in kilobytes (**RSS**), and the time or date the process was started. For instance:

```
~]$ ps aux
USER      PID %CPU %MEM      VSZ      RSS TTY      STAT START   TIME COMMAND
root      1  0.0  0.1  19404    832 ?        Ss  Mar02   0:01 /sbin/init
root      2  0.0  0.0      0      0 ?        S  Mar02   0:00 [kthreadd]
root      3  0.0  0.0      0      0 ?        S  Mar02   0:00 [migration/0]
root      4  0.0  0.0      0      0 ?        S  Mar02   0:00 [ksftirqd/0]
root      5  0.0  0.0      0      0 ?        S  Mar02   0:00 [migration/0]
root      6  0.0  0.0      0      0 ?        R  Mar02   0:00 [watchdog/0]
[output truncated]
```

You can also use the **ps** command in a combination with **grep** to see if a particular process is running. For example, to determine if **Emacs** is running, type:

```
~]$ ps ax | grep emacs
12056 pts/3    S+      0:00 emacs
12060 pts/2    S+      0:00 grep --color=auto emacs
```

For a complete list of available command line options, see the **ps(1)** manual page.

16.1.2. Using the top Command

The **top** command displays a real-time list of processes that are running on the system. It also displays additional information about the system uptime, current CPU and memory usage, or total number of running processes, and allows you to perform actions such as sorting the list or killing a process.

To run the **top** command, type the following at a shell prompt:

```
top
```

For each listed process, the **top** command displays the process ID (**PID**), the effective username of the process owner (**USER**), the priority (**PR**), the nice value (**NI**), the amount of virtual memory the process uses (**VIRT**), the amount of non-swapped physical memory the process uses (**RES**), the amount of shared memory the process uses (**SHR**), the percentage of the CPU (**%CPU**) and memory (**%MEM**) usage, the cumulated CPU time (**TIME+**), and the name of the executable file (**COMMAND**). For example:

```
~]$ top
top - 02:19:11 up 4 days, 10:37, 5 users, load average: 0.07, 0.13, 0.09
Tasks: 160 total, 1 running, 159 sleeping, 0 stopped, 0 zombie
Cpu(s): 10.7%us, 1.0%sy, 0.0%ni, 88.3%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 760752k total, 644360k used, 116392k free, 3988k buffers
Swap: 1540088k total, 76648k used, 1463440k free, 196832k cached

      PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+   COMMAND
14401 jhradile  20   0  313m   10m  5732 S  5.6  1.4  6:27.29 gnome-system-mo
  1764 root      20   0  133m   23m  4756 S  5.3  3.2  6:32.66 Xorg
13865 jhradile  20   0 1625m  177m  6628 S  0.7 23.8  0:57.26 java
  20 root      20   0     0     0 S  0.3  0.0  4:44.39 ata/0
  2085 root      20   0 40396   348   276 S  0.3  0.0  1:57.13 udisks-daemon
    1 root      20   0 19404   832   604 S  0.0  0.1  0:01.21 init
    2 root      20   0     0     0 S  0.0  0.0  0:00.01 kthreadd
    3 root      RT   0     0     0 S  0.0  0.0  0:00.00 migration/0
    4 root      20   0     0     0 S  0.0  0.0  0:00.02 ksoftirqd/0
    5 root      RT   0     0     0 S  0.0  0.0  0:00.00 migration/0
    6 root      RT   0     0     0 S  0.0  0.0  0:00.00 watchdog/0
    7 root      20   0     0     0 S  0.0  0.0  0:01.00 events/0
    8 root      20   0     0     0 S  0.0  0.0  0:00.00 cpuset
    9 root      20   0     0     0 S  0.0  0.0  0:00.00 khelper
   10 root      20   0     0     0 S  0.0  0.0  0:00.00 netns
   11 root      20   0     0     0 S  0.0  0.0  0:00.00 async/mgr
   12 root      20   0     0     0 S  0.0  0.0  0:00.00 pm
[output truncated]
```

[Table 16.1, “Interactive top commands”](#) contains useful interactive commands that you can use with **top**. For more information, see the **top(1)** manual page.

Table 16.1. Interactive top commands

Command	Description
Enter, Space	Immediately refreshes the display.

Command	Description
h, ?	Displays a help screen.
k	Kills a process. You are prompted for the process ID and the signal to send to it.
n	Changes the number of displayed processes. You are prompted to enter the number.
u	Sorts the list by user.
M	Sorts the list by memory usage.
P	Sorts the list by CPU usage.
q	Terminates the utility and returns to the shell prompt.

16.1.3. Using the System Monitor Tool

The **Processes** tab of the **System Monitor** tool allows you to view, search for, change the priority of, and kill processes from the graphical user interface.

To start the **System Monitor** tool, either select **Applications → System Tools → System Monitor** from the panel, or type **gnome-system-monitor** at a shell prompt. Then click the **Processes** tab to view the list of running processes.

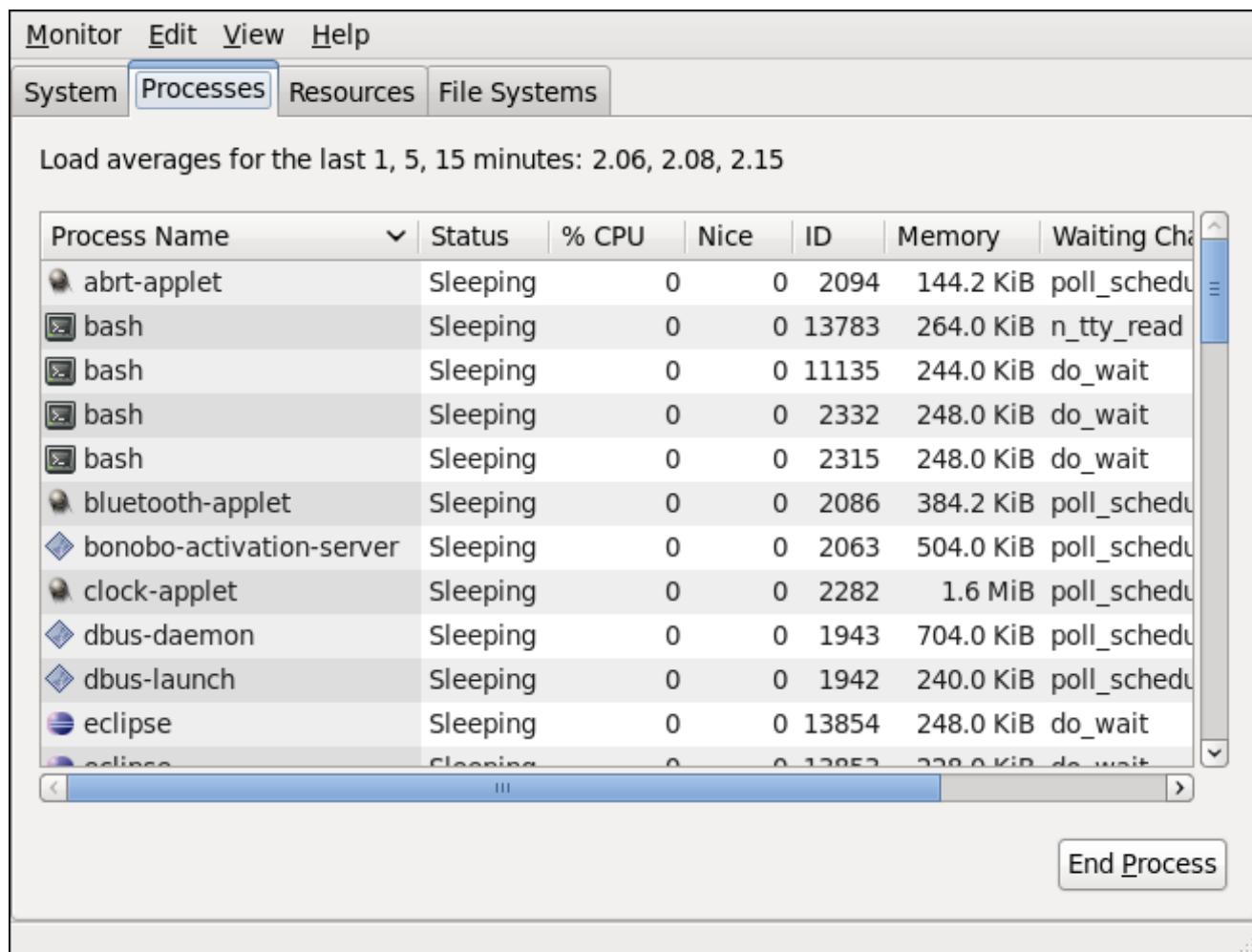


Figure 16.1. System Monitor — Processes

For each listed process, the **System Monitor** tool displays its name (**Process Name**), current status (**Status**), percentage of the CPU usage (% **CPU**), nice value (**Nice**), process ID (**ID**), memory usage (**Memory**), the channel the process is waiting in (**Waiting Channel**), and additional details about the session (**Session**). To sort the information by a specific column in ascending order, click the name of that column. Click the name of the column again to toggle the sort between ascending and descending order.

By default, the **System Monitor** tool displays a list of processes that are owned by the current user. Selecting various options from the **View** menu allows you to:

- ▶ view only active processes,
- ▶ view all processes,
- ▶ view your processes,
- ▶ view process dependencies,
- ▶ view a memory map of a selected process,
- ▶ view the files opened by a selected process, and
- ▶ refresh the list of processes.

Additionally, various options in the **Edit** menu allows you to:

- ▶ stop a process,
- ▶ continue running a stopped process,
- ▶ end a process,
- ▶ kill a process,
- ▶ change the priority of a selected process, and
- ▶ edit the **System Monitor** preferences, such as the refresh interval for the list of processes, or what information to show.

You can also end a process by selecting it from the list and clicking the **End Process** button.

16.2. Viewing Memory Usage

16.2.1. Using the free Command

The **free** command allows you to display the amount of free and used memory on the system. To do so, type the following at a shell prompt:

```
free
```

The **free** command provides information about both the physical memory (**Mem**) and swap space (**Swap**). It displays the total amount of memory (**total**), as well as the amount of memory that is in use (**used**), free (**free**), shared (**shared**), in kernel buffers (**buffers**), and cached (**cached**). For example:

```
~]$ free
              total        used        free      shared  buffers     cached
Mem:       760752      661332      99420          0       6476    317200
-/+ buffers/cache:   337656     423096
```

Swap :	1540088	283652	1256436
--------	---------	--------	---------

By default, **free** displays the values in kilobytes. To display the values in megabytes, supply the **-m** command line option:

```
free -m
```

For instance:

```
~]$ free -m
              total        used        free      shared      buffers      cached
Mem :       742         646         96          0          6        309
 -/+ buffers/cache:   330        412
Swap :      1503        276       1227
```

For a complete list of available command line options, see the **free(1)** manual page.

16.2.2. Using the System Monitor Tool

The **Resources** tab of the **System Monitor** tool allows you to view the amount of free and used memory on the system.

To start the **System Monitor** tool, either select **Applications → System Tools → System Monitor** from the panel, or type **gnome-system-monitor** at a shell prompt. Then click the **Resources** tab to view the system's memory usage.

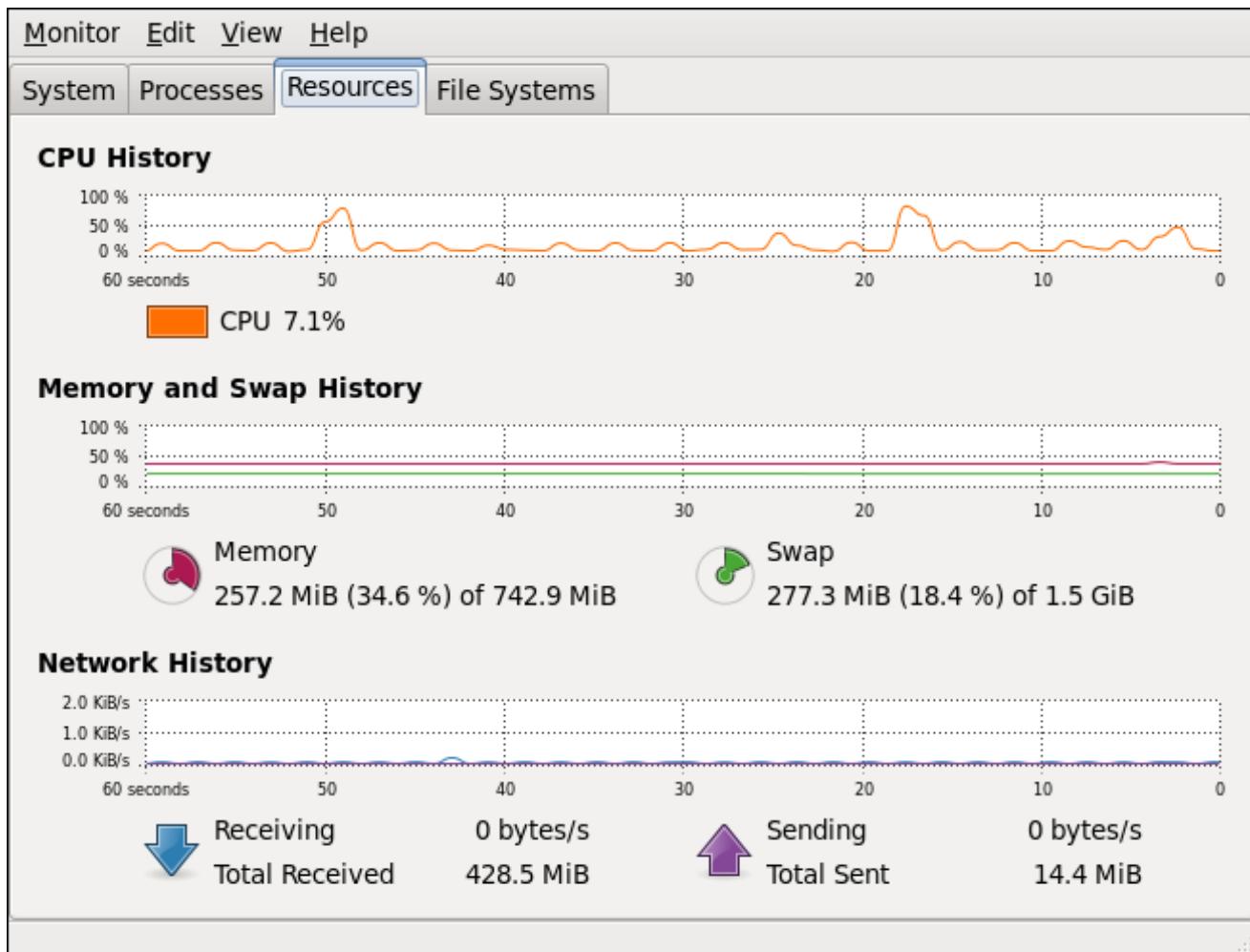


Figure 16.2. System Monitor — Resources

In the **Memory and Swap History** section, the **System Monitor** tool displays a graphical representation of the memory and swap usage history, as well as the total amount of the physical memory (**Memory**) and swap space (**Swap**) and how much of it is in use.

16.3. Viewing CPU Usage

16.3.1. Using the System Monitor Tool

The **Resources** tab of the **System Monitor** tool allows you to view the current CPU usage on the system.

To start the **System Monitor** tool, either select **Applications → System Tools → System Monitor** from the panel, or type **gnome-system-monitor** at a shell prompt. Then click the **Resources** tab to view the system's CPU usage.

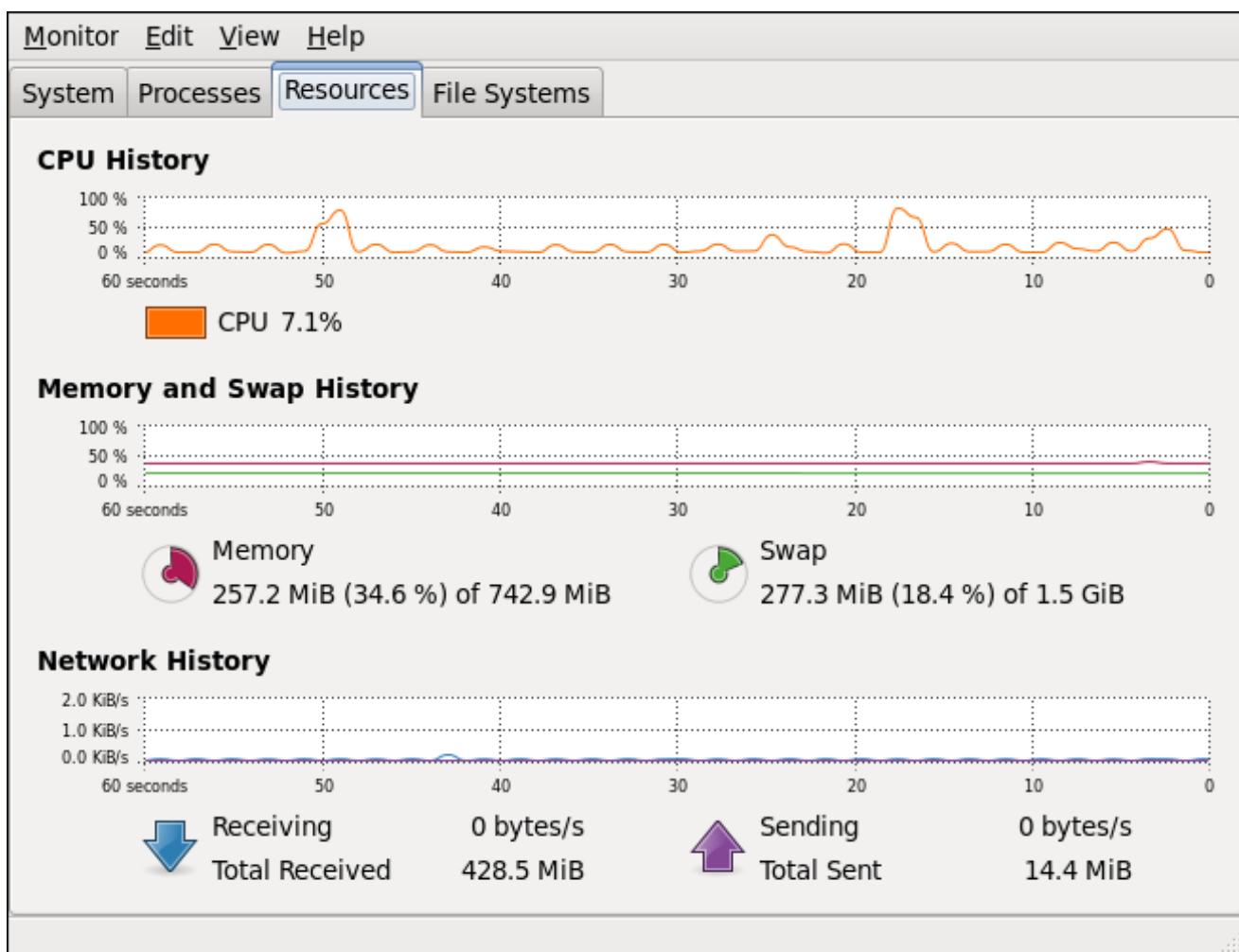


Figure 16.3. System Monitor — Resources

In the **CPU History** section, the **System Monitor** tool displays a graphical representation of the CPU usage history and shows the percentage of how much CPU is currently in use.

16.4. Viewing Block Devices and File Systems

16.4.1. Using the **lsblk** Command

The **lsblk** command allows you to display a list of available block devices. To do so, type the following at a shell prompt:

```
lsblk
```

For each listed block device, the **lsblk** command displays the device name (**NAME**), major and minor device number (**MAJ:MIN**), if the device is removable (**RM**), what is its size (**SIZE**), if the device is read-only (**RO**), what type is it (**TYPE**), and where the device is mounted (**MOUNTPOINT**). For example:

```
~]$ lsblk
NAME                MAJ:MIN RM    SIZE RO TYPE MOUNTPOINT
sr0                  11:0    1  1024M  0 rom
vda                 252:0   0   20G  0 rom
`-vda1              252:1   0   500M  0 part /boot
`-vda2              252:2   0  19.5G  0 part
  |-vg_kvm-lv_root (dm-0) 253:0   0    18G  0 lvm   /
  `|-vg_kvm-lv_swap (dm-1) 253:1   0    1.5G  0 lvm   [SWAP]
```

By default, **lsblk** lists block devices in a tree-like format. To display the information as an ordinary list, add the **-l** command line option:

```
lsblk -l
```

For instance:

```
~]$ lsblk -l
NAME                MAJ:MIN RM    SIZE RO TYPE MOUNTPOINT
sr0                  11:0    1  1024M  0 rom
vda                 252:0   0   20G  0 rom
vda1              252:1   0   500M  0 part /boot
vda2              252:2   0  19.5G  0 part
vg_kvm-lv_root (dm-0) 253:0   0    18G  0 lvm   /
vg_kvm-lv_swap (dm-1) 253:1   0    1.5G  0 lvm   [SWAP]
```

For a complete list of available command line options, see the **lsblk(8)** manual page.

16.4.2. Using the **blkid** Command

The **blkid** command allows you to display information about available block devices. To do so, type the following at a shell prompt as **root**:

```
blkid
```

For each listed block device, the **blkid** command displays available attributes such as its *universally unique identifier* (**UUID**), file system type (**TYPE**), or volume label (**LABEL**). For example:

```
~]# blkid
/dev/vda1: UUID="7fa9c421-0054-4555-b0ca-b470a97a3d84" TYPE="ext4"
/dev/vda2: UUID="7IvYzk-TnnK-oPjf-ipdD-cofz-DXaJ-gPdgBW" TYPE="LVM2_member"
/dev/mapper/vg_kvm-lv_root: UUID="a07b967c-71a0-4925-ab02-aebcad2ae824" TYPE="ext4"
/dev/mapper/vg_kvm-lv_swap: UUID="d7ef54ca-9c41-4de4-ac1b-4193b0c1ddb6"
TYPE="swap"
```

By default, the **lsblk** command lists all available block devices. To display information about a particular device only, specify the device name on the command line:

```
blkid device_name
```

For instance, to display information about **/dev/vda1**, type:

```
~]# blkid /dev/vda1
/dev/vda1: UUID="7fa9c421-0054-4555-b0ca-b470a97a3d84" TYPE="ext4"
```

You can also use the above command with the **-p** and **-o udev** command line options to obtain more detailed information. Note that **root** privileges are required to run this command:

```
blkid -po udev device_name
```

For example:

```
~]# blkid -po udev /dev/vda1
ID_FS_UUID=7fa9c421-0054-4555-b0ca-b470a97a3d84
ID_FS_UUID_ENC=7fa9c421-0054-4555-b0ca-b470a97a3d84
ID_FS_VERSION=1.0
ID_FS_TYPE=ext4
ID_FS_USAGE=filesystem
```

For a complete list of available command line options, see the **blkid(8)** manual page.

16.4.3. Using the **findmnt** Command

The **findmnt** command allows you to display a list of currently mounted file systems. To do so, type the following at a shell prompt:

```
findmnt
```

For each listed file system, the **findmnt** command displays the target mount point (**TARGET**), source device (**SOURCE**), file system type (**FSTYPE**), and relevant mount options (**OPTIONS**). For example:

```
~]$ findmnt
TARGET SOURCE FSTYPE OPTIONS
/ /dev/mapper/vg_kvm-lv_root ext4 rw,relatime,sec
|-/proc /proc proc rw,relatime
| |-/proc/bus/usb /proc/bus/usb usbfs rw,relatime
| `-/proc/sys/fs/binfmt_misc binfmt_m rw,relatime
|-/sys /sys sysfs rw,relatime,sec
|-/selinux selinuxf rw,relatime
|-/dev udev devtmpfs rw,relatime,sec
| `-dev udev devtmpfs rw,relatime,sec
|   |-/dev/pts devpts devpts rw,relatime,sec
|     `/-/dev/shm tmpfs tmpfs rw,relatime,sec
|-/boot /dev/vda1 ext4 rw,relatime,sec
|-/var/lib/nfs/rpc_pipefs sunrpc rpc_pipe rw,relatime
|-/misc /etc/auto.misc autofs rw,relatime,fd=
`-/net -hosts autofs rw,relatime,fd=
[output truncated]
```

By default, **findmnt** lists file systems in a tree-like format. To display the information as an ordinary list, add the **-l** command line option:

findmnt -l

For instance:

```
~]$ findmnt -l
TARGET SOURCE FSTYPE OPTIONS
/proc /proc proc rw,relatime
/sys /sys sysfs rw,relatime,seclabe
/dev udev devtmpfs rw,relatime,seclabe
/dev/pts devpts devpts rw,relatime,seclabe
/dev/shm tmpfs tmpfs rw,relatime,seclabe
/ /dev/mapper/vg_kvm-lv_root ext4 rw,relatime,seclabe
/selinux udev selinuxf rw,relatime
/dev /udev devtmpfs rw,relatime,seclabe
/proc/bus/usb /proc/bus/usb usbfs rw,relatime
/boot /dev/vda1 ext4 rw,relatime,seclabe
/proc/sys/fs/binfmt_misc binfmt_m_m rw,relatime
/var/lib/nfs/rpc_pipefs sunrpc rpc_pipe rw,relatime
/misc /etc/auto.misc autoofs rw,relatime,fd=7,pg
/net -hosts autoofs rw,relatime,fd=13,p
[output truncated]
```

You can also choose to list only file systems of a particular type. To do so, add the **-t** command line option followed by a file system type:

findmnt -t type

For example, to all list **ext4** file systems, type:

```
~]$ findmnt -t ext4
TARGET SOURCE FSTYPE OPTIONS
/ /dev/mapper/vg_kvm-lv_root ext4 rw,relatime,seclabel,barrier=1,data=ord
/boot /dev/vda1 ext4 rw,relatime,seclabel,barrier=1,data=ord
```

For a complete list of available command line options, see the **findmnt(8)** manual page.

16.4.4. Using the df Command

The **df** command allows you to display a detailed report on the system's disk space usage. To do so, type the following at a shell prompt:

df

For each listed file system, the **df** command displays its name (**Filesystem**), size (**1K-blocks** or **Size**), how much space is used (**Used**), how much space is still available (**Available**), the percentage of space usage (**Use%**), and where is the file system mounted (**Mounted on**). For example:

```
~]$ df
Filesystem 1K-blocks Used Available Use% Mounted on
/dev/mapper/vg_kvm-lv_root 18618236 4357360 13315112 25% /
tmpfs 380376 288 380088 1% /dev/shm
/dev/vda1 495844 77029 393215 17% /boot
```

By default, the **df** command shows the partition size in 1 kilobyte blocks and the amount of used and available disk space in kilobytes. To view the information in megabytes and gigabytes, supply the **-h** command line option, which causes **df** to display the values in a human-readable format:

df -h

For instance:

```
~]$ df -h
Filesystem           Size  Used Avail Use% Mounted on
/dev/mapper/vg_kvm-lv_root 18G  4.2G  13G  25% /
tmpfs                372M 288K  372M   1% /dev/shm
/dev/vda1              485M   76M  384M  17% /boot
```

For a complete list of available command line options, see the **df(1)** manual page.

16.4.5. Using the du Command

The **du** command allows you to displays the amount of space that is being used by files in a directory. To display the disk usage for each of the subdirectories in the current working directory, run the command with no additional command line options:

du

For example:

```
~]$ du
14972 ./Downloads
4      ./gnome2
4      ./mozilla/extensions
4      ./mozilla/plugins
12     ./mozilla
15004 .
```

By default, the **du** command displays the disk usage in kilobytes. To view the information in megabytes and gigabytes, supply the **-h** command line option, which causes the utility to display the values in a human-readable format:

du -h

For instance:

```
~]$ du -h
15M    ./Downloads
4.0K   ./gnome2
4.0K   ./mozilla/extensions
4.0K   ./mozilla/plugins
12K   ./mozilla
15M   .
```

At the end of the list, the **du** command always shows the grand total for the current directory. To display only this information, supply the **-s** command line option:

du -sh

For example:

```
~]$ du -sh
15M   .
```

For a complete list of available command line options, see the **du(1)** manual page.

16.4.6. Using the System Monitor Tool

The **File Systems** tab of the **System Monitor** tool allows you to view file systems and disk space usage in the graphical user interface.

To start the **System Monitor** tool, either select **Applications → System Tools → System Monitor** from the panel, or type **gnome-system-monitor** at a shell prompt. Then click the **File Systems** tab to view a list of file systems.

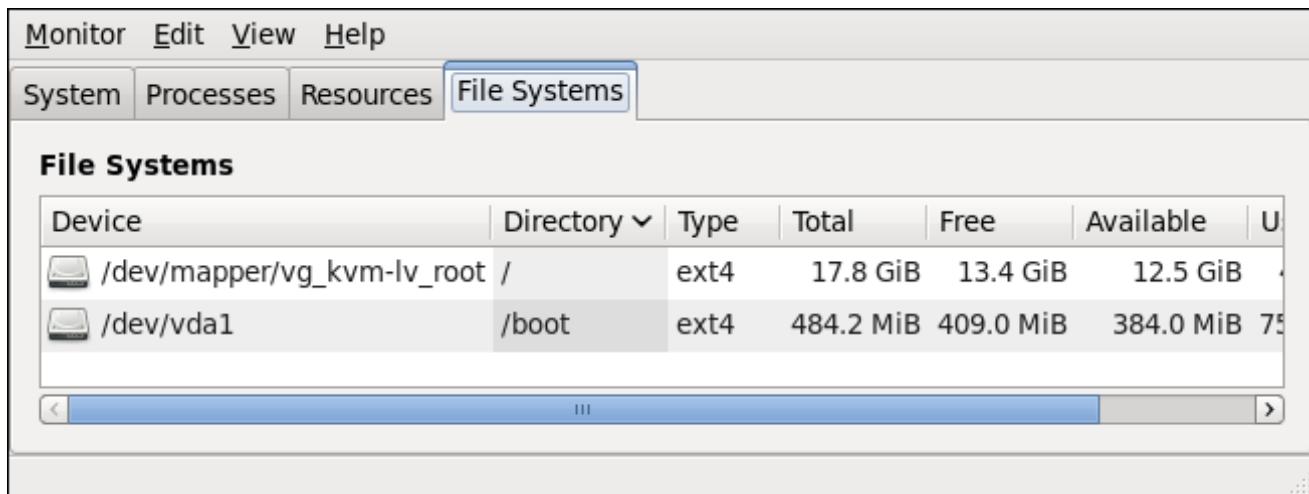


Figure 16.4. System Monitor — File Systems

For each listed file system, the **System Monitor** tool displays the source device (**Device**), target mount point (**Directory**), and file system type (**Type**), as well as its size (**Total**) and how much space is free (**Free**), available (**Available**), and used (**Used**).

16.5. Viewing Hardware Information

16.5.1. Using the lspci Command

The **lspci** command allows you to display information about PCI buses and devices that are attached to them. To list all PCI devices that are in the system, type the following at a shell prompt:

```
lspci
```

This displays a simple list of devices, for example:

```
~]$ lspci
00:00.0 Host bridge: Intel Corporation 82X38/X48 Express DRAM Controller
00:01.0 PCI bridge: Intel Corporation 82X38/X48 Express Host-Primary PCI Express
Bridge
00:1a.0 USB Controller: Intel Corporation 82801I (ICH9 Family) USB UHCI Controller
#4 (rev 02)
00:1a.1 USB Controller: Intel Corporation 82801I (ICH9 Family) USB UHCI Controller
#5 (rev 02)
00:1a.2 USB Controller: Intel Corporation 82801I (ICH9 Family) USB UHCI Controller
#6 (rev 02)
[output truncated]
```

You can also use the **-v** command line option to display more verbose output, or **-vv** for very verbose output:

```
lspci -v| -vv
```

For instance, to determine the manufacturer, model, and memory size of a system's video card, type:

```
~]$ lspci -v  
[output truncated]
```

```
01:00.0 VGA compatible controller: nVidia Corporation G84 [Quadro FX 370] (rev a1)
  (prog-if 00 [VGA controller])
    Subsystem: nVidia Corporation Device 0491
    Physical Slot: 2
    Flags: bus master, fast devsel, latency 0, IRQ 16
    Memory at f2000000 (32-bit, non-prefetchable) [size=16M]
    Memory at e0000000 (64-bit, prefetchable) [size=256M]
    Memory at f0000000 (64-bit, non-prefetchable) [size=32M]
    I/O ports at 1100 [size=128]
    Expansion ROM at <unassigned> [disabled]
    Capabilities: <access denied>
    Kernel driver in use: nouveau
    Kernel modules: nouveau, nvidiafb
```

```
[output truncated]
```

For a complete list of available command line options, see the **[lspci\(8\)](#)** manual page.

16.5.2. Using the **lsusb** Command

The **lsusb** command allows you to display information about USB buses and devices that are attached to them. To list all USB devices that are in the system, type the following at a shell prompt:

```
lsusb
```

This displays a simple list of devices, for example:

```
~]$ lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
[output truncated]
Bus 001 Device 002: ID 0bda:0151 Realtek Semiconductor Corp. Mass Storage Device
(Multicard Reader)
Bus 008 Device 002: ID 03f0:2c24 Hewlett-Packard Logitech M-UAL-96 Mouse
Bus 008 Device 003: ID 04b3:3025 IBM Corp.
```

You can also use the **-v** command line option to display more verbose output:

```
lsusb -v
```

For instance:

```
~]$ lsusb -v  
[output truncated]
```

```
Bus 008 Device 002: ID 03f0:2c24 Hewlett-Packard Logitech M-UAL-96 Mouse
Device Descriptor:
```

```

bLength          18
bDescriptorType 1
bcdUSB          2.00
bDeviceClass     0 (Defined at Interface level)
bDeviceSubClass  0
bDeviceProtocol 0
bMaxPacketSize0 8
idVendor         0x03f0 Hewlett-Packard
idProduct        0x2c24 Logitech M-UAL-96 Mouse
bcdDevice        31.00
iManufacturer    1
iProduct         2
iSerial          0
bNumConfigurations 1
Configuration Descriptor:
  bLength          9
  bDescriptorType  2
[output truncated]

```

For a complete list of available command line options, see the **lsubusb(8)** manual page.

16.5.3. Using the **lspcmcia** Command

The **lspcmcia** command allows you to list all PCMCIA devices that are present in the system. To do so, type the following at a shell prompt:

```
lspcmcia
```

For example:

```

~]$ lspcmcia
Socket 0 Bridge:      [yenta_cardbus]           (bus ID: 0000:15:00.0)

```

You can also use the **-v** command line option to display more verbose information, or **-vv** to increase the verbosity level even further:

```
lspcmcia -v|-vv
```

For instance:

```

~]$ lspcmcia -v
Socket 0 Bridge:      [yenta_cardbus]           (bus ID: 0000:15:00.0)
  Configuration: state: on       ready: unknown

```

For a complete list of available command line options, see the **pccardctl(8)** manual page.

16.5.4. Using the **lscpu** Command

The **lscpu** command allows you to list information about CPUs that are present in the system, including the number of CPUs, their architecture, vendor, family, model, CPU caches, etc. To do so, type the following at a shell prompt:

```
lscpu
```

For example:

```
~]$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:             Little Endian
CPU(s):                4
On-line CPU(s) list:   0-3
Thread(s) per core:    1
Core(s) per socket:    4
Socket(s):              1
NUMA node(s):           1
Vendor ID:              GenuineIntel
CPU family:             6
Model:                 23
Stepping:               7
CPU MHz:                1998.000
BogoMIPS:               4999.98
Virtualization:         VT-x
L1d cache:              32K
L1i cache:              32K
L2 cache:                3072K
NUMA node0 CPU(s):      0-3
```

For a complete list of available command line options, see the **`lscpu(1)`** manual page.

16.6. Monitoring Performance with Net-SNMP

Red Hat Enterprise Linux 7 includes the **Net-SNMP** software suite, which includes a flexible and extensible *Simple Network Management Protocol* (SNMP) agent. This agent and its associated utilities can be used to provide performance data from a large number of systems to a variety of tools which support polling over the SNMP protocol.

This section provides information on configuring the Net-SNMP agent to securely provide performance data over the network, retrieving the data using the SNMP protocol, and extending the SNMP agent to provide custom performance metrics.

16.6.1. Installing Net-SNMP

The Net-SNMP software suite is available as a set of RPM packages in the Red Hat Enterprise Linux software distribution. [Table 16.2, “Available Net-SNMP packages”](#) summarizes each of the packages and their contents.

Table 16.2. Available Net-SNMP packages

Package	Provides
<code>net-snmp</code>	The SNMP Agent Daemon and documentation. This package is required for exporting performance data.
<code>net-snmp-libs</code>	The netsnmp library and the bundled management information bases (MIBs). This package is required for exporting performance data.
<code>net-snmp-utils</code>	SNMP clients such as snmpget and snmpwalk . This package is required in order to query a system's performance data over SNMP.

To install any of these packages, use the **`yum`** command in the following form:

```
yum install package...
```

For example, to install the SNMP Agent Daemon and SNMP clients used in the rest of this section, type the following at a shell prompt:

```
~]# yum install net-snmp net-snmp-libs net-snmp-utils
```

Note that you must have superuser privileges (that is, you must be logged in as **root**) to run this command. For more information on how to install new packages in Red Hat Enterprise Linux, see [Section 5.2.4, “Installing Packages”](#).

16.6.2. Running the Net-SNMP Daemon

The *net-snmp* package contains **snmpd**, the SNMP Agent Daemon. This section provides information on how to start, stop, and restart the **snmpd** service, and shows how to enable it in a particular runlevel. For more information on the concept of runlevels and how to manage system services in Red Hat Enterprise Linux in general, see [Chapter 6, Managing Services with systemd](#).

16.6.2.1. Starting the Service

To run the **snmpd** service in the current session, type the following at a shell prompt as **root**:

```
systemctl start snmpd.service
```

To configure the service to be automatically started at boot time, use the following command:

```
systemctl enable snmpd.service
```

16.6.2.2. Stopping the Service

To stop the running **snmpd** service, type the following at a shell prompt as **root**:

```
systemctl stop snmpd.service
```

To disable starting the service at boot time, use the following command:

```
systemctl disable snmpd.service
```

16.6.2.3. Restarting the Service

To restart the running **snmpd** service, type the following at a shell prompt:

```
systemctl restart snmpd.service
```

This command stops the service and starts it again in quick succession. To only reload the configuration without stopping the service, run the following command instead:

```
systemctl reload snmpd.service
```

This causes the running **snmpd** service to reload its configuration.

16.6.3. Configuring Net-SNMP

To change the Net-SNMP Agent Daemon configuration, edit the `/etc/snmp/snmpd.conf` configuration file. The default `snmpd.conf` file shipped with Red Hat Enterprise Linux 7 is heavily commented and serves as a good starting point for agent configuration.

This section focuses on two common tasks: setting system information and configuring authentication. For more information about available configuration directives, see the `snmpd.conf(5)` manual page. Additionally, there is a utility in the `net-snmp` package named `snmpconf` which can be used to interactively generate a valid agent configuration.

Note that the `net-snmp-utils` package must be installed in order to use the `snmpwalk` utility described in this section.



Applying the changes

For any changes to the configuration file to take effect, force the `snmpd` service to re-read the configuration by running the following command as `root`:

```
systemctl reload snmpd.service
```

16.6.3.1. Setting System Information

Net-SNMP provides some rudimentary system information via the `system` tree. For example, the following `snmpwalk` command shows the `system` tree with a default agent configuration.

```
~]# snmpwalk -v2c -c public localhost system
SNMPv2-MIB::sysDescr.0 = STRING: Linux localhost.localdomain 2.6.32-122.el6.x86_64 #1
SMP Wed Mar 9 23:54:34 EST 2011 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (99554) 0:16:35.54
SNMPv2-MIB::sysContact.0 = STRING: Root <root@localhost> (configure
/etc/snmp/snmp.local.conf)
SNMPv2-MIB::sysName.0 = STRING: localhost.localdomain
SNMPv2-MIB::sysLocation.0 = STRING: Unknown (edit /etc/snmp/snmpd.conf)
```

By default, the `sysName` object is set to the hostname. The `sysLocation` and `sysContact` objects can be configured in the `/etc/snmp/snmpd.conf` file by changing the value of the `syslocation` and `syscontact` directives, for example:

```
syslocation Datacenter, Row 3, Rack 2
syscontact UNIX Admin <admin@example.com>
```

After making changes to the configuration file, reload the configuration and test it by running the `snmpwalk` command again:

```
~]# systemctl reload snmp.service
~]# snmpwalk -v2c -c public localhost system
SNMPv2-MIB::sysDescr.0 = STRING: Linux localhost.localdomain 2.6.32-122.el6.x86_64 #1
SMP Wed Mar 9 23:54:34 EST 2011 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (158357) 0:26:23.57
SNMPv2-MIB::sysContact.0 = STRING: UNIX Admin <admin@example.com>
SNMPv2-MIB::sysName.0 = STRING: localhost.localdomain
SNMPv2-MIB::sysLocation.0 = STRING: Datacenter, Row 3, Rack 2
```

16.6.3.2. Configuring Authentication

The Net-SNMP Agent Daemon supports all three versions of the SNMP protocol. The first two versions (1 and 2c) provide for simple authentication using a *community string*. This string is a shared secret between the agent and any client utilities. The string is passed in clear text over the network however and is not considered secure. Version 3 of the SNMP protocol supports user authentication and message encryption using a variety of protocols. The Net-SNMP agent also supports tunneling over SSH, TLS authentication with X.509 certificates, and Kerberos authentication.

Configuring SNMP Version 2c Community

To configure an **SNMP version 2c community**, use either the **rocommunity** or **rwcommunity** directive in the **/etc/snmp/snmpd.conf** configuration file. The format of the directives is the following:

```
directive community [source [OID]]
```

... where *community* is the community string to use, *source* is an IP address or subnet, and *OID* is the SNMP tree to provide access to. For example, the following directive provides read-only access to the **system** tree to a client using the community string “redhat” on the local machine:

```
rocommunity redhat 127.0.0.1 .1.3.6.1.2.1.1
```

To test the configuration, use the **snmpwalk** command with the **-v** and **-c** options.

```
~]# snmpwalk -v2c -c redhat localhost system
SNMPv2-MIB::sysDescr.0 = STRING: Linux localhost.localdomain 2.6.32-122.el6.x86_64 #1
SMP Wed Mar 9 23:54:34 EST 2011 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (158357) 0:26:23.57
SNMPv2-MIB::sysContact.0 = STRING: UNIX Admin <admin@example.com>
SNMPv2-MIB::sysName.0 = STRING: localhost.localdomain
SNMPv2-MIB::sysLocation.0 = STRING: Datacenter, Row 3, Rack 2
```

Configuring SNMP Version 3 User

To configure an **SNMP version 3 user**, use the **net-snmp-create-v3-user** command. This command adds entries to the **/var/lib/net-snmp/snmpd.conf** and **/etc/snmp/snmpd.conf** files which create the user and grant access to the user. Note that the **net-snmp-create-v3-user** command may only be run when the agent is not running. The following example creates the “admin” user with the password “redhatsnmp”:

```
~]# systemctl stop snmpd.service
~]# net-snmp-create-v3-user
Enter a SNMPv3 user name to create:
admin
Enter authentication pass-phrase:
redhatsnmp
Enter encryption pass-phrase:
[press return to reuse the authentication pass-phrase]

adding the following line to /var/lib/net-snmp/snmpd.conf:
  createUser admin MD5 "redhatsnmp" DES
adding the following line to /etc/snmp/snmpd.conf:
  rwuser admin
~]# systemctl start snmpd.service
```

The **rwuser** directive (or **rouser** when the **-ro** command line option is supplied) that **net-snmp-create-v3-user** adds to **/etc/snmp/snmpd.conf** has a similar format to the **rwcommunity** and **rocommunity** directives:

```
directive user [noauth|auth|priv] [OID]
```

... where *user* is a username and *OID* is the SNMP tree to provide access to. By default, the Net-SNMP Agent Daemon allows only authenticated requests (the **auth** option). The **noauth** option allows you to permit unauthenticated requests, and the **priv** option enforces the use of encryption. The **authpriv** option specifies that requests must be authenticated and replies should be encrypted.

For example, the following line grants the user “admin” read-write access to the entire tree:

```
rwuser admin authpriv .1
```

To test the configuration, create a **.snmp** directory in your user’s home directory and a configuration file named **snmp.conf** in that directory (**~/.snmp/snmp.conf**) with the following lines:

```
defVersion 3
defSecurityLevel authPriv
defSecurityName admin
defPassphrase redhatsnmp
```

The **snmpwalk** command will now use these authentication settings when querying the agent:

```
~]$ snmpwalk -v3 localhost system
SNMPv2-MIB::sysDescr.0 = STRING: Linux localhost.localdomain 2.6.32-122.el6.x86_64 #1
SMP Wed Mar 9 23:54:34 EST 2011 x86_64
[output truncated]
```

16.6.4. Retrieving Performance Data over SNMP

The Net-SNMP Agent in Red Hat Enterprise Linux provides a wide variety of performance information over the SNMP protocol. In addition, the agent can be queried for a listing of the installed RPM packages on the system, a listing of currently running processes on the system, or the network configuration of the system.

This section provides an overview of OIDs related to performance tuning available over SNMP. It assumes that the **net-snmp-utils** package is installed and that the user is granted access to the SNMP tree as described in [Section 16.6.3.2, “Configuring Authentication”](#).

16.6.4.1. Hardware Configuration

The **Host Resources MIB** included with Net-SNMP presents information about the current hardware and software configuration of a host to a client utility. [Table 16.3, “Available OIDs”](#) summarizes the different OIDs available under that MIB.

Table 16.3. Available OIDs

OID	Description
HOST -RESOURCES-MIB::hrSystem	Contains general system information such as uptime, number of users, and number of running processes.
HOST -RESOURCES-MIB::hrStorage	Contains data on memory and file system usage.

OID	Description
HOST -RESOURCES-MIB::hrDevices	Contains a listing of all processors, network devices, and file systems.
HOST -RESOURCES-MIB::hrSWRun	Contains a listing of all running processes.
HOST -RESOURCES-MIB::hrSWRunPerf	Contains memory and CPU statistics on the process table from HOST-RESOURCES-MIB::hrSWRun.
HOST -RESOURCES-MIB::hrSWInstalled	Contains a listing of the RPM database.

There are also a number of SNMP tables available in the Host Resources MIB which can be used to retrieve a summary of the available information. The following example displays **HOST -RESOURCES-MIB::hrFSTable**:

```
~]$ snmpstable -Cb localhost HOST-RESOURCES-MIB::hrFSTable
SNMP table: HOST-RESOURCES-MIB::hrFSTable

Index MountPoint RemoteMountPoint Type
Access Bootable StorageIndex LastFullBackupDate LastPartialBackupDate
    1      "/"           ""   HOST-RESOURCES-TYPES::hrFSLinuxExt2
readWrite  true        31   0-1-1,0:0:0.0  0-1-1,0:0:0.0
      5  "/dev/shm"     ""   HOST-RESOURCES-TYPES::hrFSOther
readWrite  false       35   0-1-1,0:0:0.0  0-1-1,0:0:0.0
      6  "/boot"        ""   HOST-RESOURCES-TYPES::hrFSLinuxExt2
readWrite  false       36   0-1-1,0:0:0.0  0-1-1,0:0:0.0
```

For more information about **HOST -RESOURCES-MIB**, see the **/usr/share/snmp/mibs/HOST-RESOURCES-MIB.txt** file.

16.6.4.2. CPU and Memory Information

Most system performance data is available in the **UCD SNMP MIB**. The **systemStats** OID provides a number of counters around processor usage:

```
~]$ snmpwalk localhost UCD-SNMP-MIB::systemStats
UCD-SNMP-MIB::ssIndex.0 = INTEGER: 1
UCD-SNMP-MIB::ssErrorName.0 = STRING: systemStats
UCD-SNMP-MIB::ssSwapIn.0 = INTEGER: 0 kB
UCD-SNMP-MIB::ssSwapOut.0 = INTEGER: 0 kB
UCD-SNMP-MIB::ssIOSent.0 = INTEGER: 0 blocks/s
UCD-SNMP-MIB::ssIOReceive.0 = INTEGER: 0 blocks/s
UCD-SNMP-MIB::ssSysInterrupts.0 = INTEGER: 29 interrupts/s
UCD-SNMP-MIB::ssSysContext.0 = INTEGER: 18 switches/s
UCD-SNMP-MIB::ssCpuUser.0 = INTEGER: 0
UCD-SNMP-MIB::ssCpuSystem.0 = INTEGER: 0
UCD-SNMP-MIB::ssCpuIdle.0 = INTEGER: 99
UCD-SNMP-MIB::ssCpuRawUser.0 = Counter32: 2278
UCD-SNMP-MIB::ssCpuRawNice.0 = Counter32: 1395
UCD-SNMP-MIB::ssCpuRawSystem.0 = Counter32: 6826
UCD-SNMP-MIB::ssCpuRawIdle.0 = Counter32: 3383736
UCD-SNMP-MIB::ssCpuRawWait.0 = Counter32: 7629
UCD-SNMP-MIB::ssCpuRawKernel.0 = Counter32: 0
UCD-SNMP-MIB::ssCpuRawInterrupt.0 = Counter32: 434
UCD-SNMP-MIB::ssIOSent.0 = Counter32: 266770
UCD-SNMP-MIB::ssIORawReceived.0 = Counter32: 427302
UCD-SNMP-MIB::ssRawInterrupts.0 = Counter32: 743442
```

```
UCD-SNMP-MIB::ssRawContexts.0 = Counter32: 718557
UCD-SNMP-MIB::ssCpuRawSoftIRQ.0 = Counter32: 128
UCD-SNMP-MIB::ssRawSwapIn.0 = Counter32: 0
UCD-SNMP-MIB::ssRawSwapOut.0 = Counter32: 0
```

In particular, the **ssCpuRawUser**, **ssCpuRawSystem**, **ssCpuRawWait**, and **ssCpuRawIdle** OIDs provide counters which are helpful when determining whether a system is spending most of its processor time in kernel space, user space, or I/O. **ssRawSwapIn** and **ssRawSwapOut** can be helpful when determining whether a system is suffering from memory exhaustion.

More memory information is available under the **UCD-SNMP-MIB::memory** OID, which provides similar data to the **free** command:

```
~]$ snmpwalk localhost UCD-SNMP-MIB::memory
UCD-SNMP-MIB::memIndex.0 = INTEGER: 0
UCD-SNMP-MIB::memErrorName.0 = STRING: swap
UCD-SNMP-MIB::memTotalSwap.0 = INTEGER: 1023992 kB
UCD-SNMP-MIB::memAvailSwap.0 = INTEGER: 1023992 kB
UCD-SNMP-MIB::memTotalReal.0 = INTEGER: 1021588 kB
UCD-SNMP-MIB::memAvailReal.0 = INTEGER: 634260 kB
UCD-SNMP-MIB::memTotalFree.0 = INTEGER: 1658252 kB
UCD-SNMP-MIB::memMinimumSwap.0 = INTEGER: 16000 kB
UCD-SNMP-MIB::memBuffer.0 = INTEGER: 30760 kB
UCD-SNMP-MIB::memCached.0 = INTEGER: 216200 kB
UCD-SNMP-MIB::memSwapError.0 = INTEGER: noError(0)
UCD-SNMP-MIB::memSwapErrorMsg.0 = STRING:
```

Load averages are also available in the **UCD-SNMP-MIB**. The SNMP table **UCD-SNMP-MIB::laTable** has a listing of the 1, 5, and 15 minute load averages:

```
~]$ snmpstable localhost UCD-SNMP-MIB::laTable
SNMP table: UCD-SNMP-MIB::laTable

laIndex laNames laLoad laConfig laLoadInt laLoadFloat laErrorFlag laErrMessage
      1 Load-1   0.00    12.00        0    0.000000    noError
      2 Load-5   0.00    12.00        0    0.000000    noError
      3 Load-15  0.00    12.00        0    0.000000    noError
```

16.6.4.3. File System and Disk Information

The **Host Resources MIB** provides information on file system size and usage. Each file system (and also each memory pool) has an entry in the **HOST-RESOURCES-MIB::hrStorageTable** table:

```
~]$ snmpstable -Cb localhost HOST-RESOURCES-MIB::hrStorageTable
SNMP table: HOST-RESOURCES-MIB::hrStorageTable

  Index          Type          Descr
AllocationUnits  Size  Used AllocationFailures
      1           HOST-RESOURCES-TYPES::hrStorageRam Physical memory
1024 Bytes 1021588 388064          ?
      3 HOST-RESOURCES-TYPES::hrStorageVirtualMemory Virtual memory
1024 Bytes 2045580 388064          ?
      6           HOST-RESOURCES-TYPES::hrStorageOther Memory buffers
1024 Bytes 1021588 31048          ?
      7           HOST-RESOURCES-TYPES::hrStorageOther Cached memory
1024 Bytes 216604 216604          ?
      10          HOST-RESOURCES-TYPES::hrStorageVirtualMemory Swap space
1024 Bytes 1023992 0          ?
      31          HOST-RESOURCES-TYPES::hrStorageFixedDisk /
```

```
4096 Bytes 2277614 250391 ?  

  35      HOST-RESOURCES-TYPES::hrStorageFixedDisk /dev/shm  

4096 Bytes 127698   0   ?  

  36      HOST-RESOURCES-TYPES::hrStorageFixedDisk /boot  

1024 Bytes 198337  26694 ?
```

The OIDs under **HOST-RESOURCES-MIB::hrStorageSize** and **HOST-RESOURCES-MIB::hrStorageUsed** can be used to calculate the remaining capacity of each mounted file system.

I/O data is available both in **UCD-SNMP-MIB::systemStats** (**ssIORawSent.0** and **ssIORawReceived.0**) and in **UCD-DISKIO-MIB::diskIOTable**. The latter provides much more granular data. Under this table are OIDs for **diskIONReadX** and **diskIONWrittenX**, which provide counters for the number of bytes read from and written to the block device in question since the system boot:

```
~]$ snmpstable -Cb localhost UCD-DISKIO-MIB::diskIOTable  

SNMP table: UCD-DISKIO-MIB::diskIOTable  

Index Device      NRead  NWritten Reads Writes LA1  LA5  LA15     NReadX NWrittenX  

...  

25    sda  216886272 139109376 16409   4894   ?   ?   ? 216886272 139109376  

26    sda1 2455552      5120    613     2   ?   ?   ? 2455552      5120  

27    sda2 1486848      0     332     0   ?   ?   ? 1486848      0  

28    sda3 212321280 139104256 15312   4871   ?   ?   ? 212321280 139104256
```

16.6.4.4. Network Information

The **Interfaces MIB** provides information on network devices. **IF-MIB::ifTable** provides an SNMP table with an entry for each interface on the system, the configuration of the interface, and various packet counters for the interface. The following example shows the first few columns of **ifTable** on a system with two physical network interfaces:

```
~]$ snmpstable -Cb localhost IF-MIB::ifTable  

SNMP table: IF-MIB::ifTable  

Index Descr          Type   Mtu   Speed      PhysAddress AdminStatus  

  1   lo  softwareLoopback 16436 100000000          up  

  2   eth0  ethernetCsmacd 1500      0 52:54:0:c7:69:58          up  

  3   eth1  ethernetCsmacd 1500      0 52:54:0:a7:a3:24        down
```

Network traffic is available under the OIDs **IF-MIB::ifOutOctets** and **IF-MIB::ifInOctets**. The following SNMP queries will retrieve network traffic for each of the interfaces on this system:

```
~]$ snmpwalk localhost IF-MIB::ifDescr  

IF-MIB::ifDescr.1 = STRING: lo  

IF-MIB::ifDescr.2 = STRING: eth0  

IF-MIB::ifDescr.3 = STRING: eth1  

~]$ snmpwalk localhost IF-MIB::ifOutOctets  

IF-MIB::ifOutOctets.1 = Counter32: 10060699  

IF-MIB::ifOutOctets.2 = Counter32: 650  

IF-MIB::ifOutOctets.3 = Counter32: 0  

~]$ snmpwalk localhost IF-MIB::ifInOctets  

IF-MIB::ifInOctets.1 = Counter32: 10060699  

IF-MIB::ifInOctets.2 = Counter32: 78650  

IF-MIB::ifInOctets.3 = Counter32: 0
```

16.6.5. Extending Net-SNMP

The Net-SNMP Agent can be extended to provide application metrics in addition to raw system metrics. This allows for capacity planning as well as performance issue troubleshooting. For example, it may be helpful to know that an email system had a 5-minute load average of 15 while being tested, but it is more helpful to know that the email system has a load average of 15 while processing 80,000 messages a second. When application metrics are available via the same interface as the system metrics, this also allows for the visualization of the impact of different load scenarios on system performance (for example, each additional 10,000 messages increases the load average linearly until 100,000).

A number of the applications included in Red Hat Enterprise Linux extend the Net-SNMP Agent to provide application metrics over SNMP. There are several ways to extend the agent for custom applications as well. This section describes extending the agent with shell scripts. It assumes that the *net-snmp-utils* package is installed, and that the user is granted access to the SNMP tree as described in [Section 16.6.3.2, “Configuring Authentication”](#).

16.6.5.1. Extending Net-SNMP with Shell Scripts

The Net-SNMP Agent provides an extension MIB (**NET -SNMP -EXTEND -MIB**) that can be used to query arbitrary shell scripts. To specify the shell script to run, use the **extend** directive in the */etc/snmp/snmpd.conf* file. Once defined, the Agent will provide the exit code and any output of the command over SNMP. The example below demonstrates this mechanism with a script which determines the number of **httpd** processes in the process table.



Using the proc directive

The Net-SNMP Agent also provides a built-in mechanism for checking the process table via the **proc** directive. See the **snmpd.conf(5)** manual page for more information.

The exit code of the following shell script is the number of **httpd** processes running on the system at a given point in time:

```
#!/bin/sh
NUMPIDS=`pgrep httpd | wc -l`
exit $NUMPIDS
```

To make this script available over SNMP, copy the script to a location on the system path, set the executable bit, and add an **extend** directive to the */etc/snmp/snmpd.conf* file. The format of the **extend** directive is the following:

```
extend name prog args
```

... where *name* is an identifying string for the extension, *prog* is the program to run, and *args* are the arguments to give the program. For instance, if the above shell script is copied to **/usr/local/bin/check_apache.sh**, the following directive will add the script to the SNMP tree:

```
extend httpd_pids /bin/sh /usr/local/bin/check_apache.sh
```

The script can then be queried at **NET -SNMP -EXTEND -MIB : :nsExtendObjects**:

```
~]$ snmpwalk localhost NET -SNMP -EXTEND -MIB : :nsExtendObjects
NET -SNMP -EXTEND -MIB : :nsExtendNumEntries.0 = INTEGER: 1
NET -SNMP -EXTEND -MIB : :nsExtendCommand."httpd_pids" = STRING: /bin/sh
NET -SNMP -EXTEND -MIB : :nsExtendArgs."httpd_pids" = STRING:
```

```
/usr/local/bin/check_apache.sh
NET-SNMP-EXTEND-MIB::nsExtendInput."httpd_pids" = STRING:
NET-SNMP-EXTEND-MIB::nsExtendCacheTime."httpd_pids" = INTEGER: 5
NET-SNMP-EXTEND-MIB::nsExtendExecType."httpd_pids" = INTEGER: exec(1)
NET-SNMP-EXTEND-MIB::nsExtendRunType."httpd_pids" = INTEGER: run-on-read(1)
NET-SNMP-EXTEND-MIB::nsExtendStorage."httpd_pids" = INTEGER: permanent(4)
NET-SNMP-EXTEND-MIB::nsExtendStatus."httpd_pids" = INTEGER: active(1)
NET-SNMP-EXTEND-MIB::nsExtendOutput1Line."httpd_pids" = STRING:
NET-SNMP-EXTEND-MIB::nsExtendOutputFull."httpd_pids" = STRING:
NET-SNMP-EXTEND-MIB::nsExtendOutNumLines."httpd_pids" = INTEGER: 1
NET-SNMP-EXTEND-MIB::nsExtendResult."httpd_pids" = INTEGER: 8
NET-SNMP-EXTEND-MIB::nsExtendOutLine."httpd_pids".1 = STRING:
```

Note that the exit code ("8" in this example) is provided as an INTEGER type and any output is provided as a STRING type. To expose multiple metrics as integers, supply different arguments to the script using the **extend** directive. For example, the following shell script can be used to determine the number of processes matching an arbitrary string, and will also output a text string giving the number of processes:

```
#!/bin/sh

PATTERN=$1
NUMPIDS=`pgrep $PATTERN | wc -l`

echo "There are $NUMPIDS $PATTERN processes."
exit $NUMPIDS
```

The following **/etc/snmp/snmpd.conf** directives will give both the number of **httpd** PIDs as well as the number of **snmpd** PIDs when the above script is copied to **/usr/local/bin/check_proc.sh**:

```
extend httpd_pids /bin/sh /usr/local/bin/check_proc.sh httpd
extend snmpd_pids /bin/sh /usr/local/bin/check_proc.sh snmpd
```

The following example shows the output of an **snmpwalk** of the **nsExtendObjects** OID:

```
[~]$ snmpwalk localhost NET-SNMP-EXTEND-MIB::nsExtendObjects
NET-SNMP-EXTEND-MIB::nsExtendNumEntries.0 = INTEGER: 2
NET-SNMP-EXTEND-MIB::nsExtendCommand."httpd_pids" = STRING: /bin/sh
NET-SNMP-EXTEND-MIB::nsExtendCommand."snmpd_pids" = STRING: /bin/sh
NET-SNMP-EXTEND-MIB::nsExtendArgs."httpd_pids" = STRING:
/usr/local/bin/check_proc.sh httpd
NET-SNMP-EXTEND-MIB::nsExtendArgs."snmpd_pids" = STRING:
/usr/local/bin/check_proc.sh snmpd
NET-SNMP-EXTEND-MIB::nsExtendInput."httpd_pids" = STRING:
NET-SNMP-EXTEND-MIB::nsExtendInput."snmpd_pids" = STRING:
...
NET-SNMP-EXTEND-MIB::nsExtendResult."httpd_pids" = INTEGER: 8
NET-SNMP-EXTEND-MIB::nsExtendResult."snmpd_pids" = INTEGER: 1
NET-SNMP-EXTEND-MIB::nsExtendOutLine."httpd_pids".1 = STRING: There are 8 httpd
processes.
NET-SNMP-EXTEND-MIB::nsExtendOutLine."snmpd_pids".1 = STRING: There are 1 snmpd
processes.
```



Integer exit codes are limited

Integer exit codes are limited to a range of 0–255. For values that are likely to exceed 256, either use the standard output of the script (which will be typed as a string) or a different method of extending the agent.

This last example shows a query for the free memory of the system and the number of **httpd** processes. This query could be used during a performance test to determine the impact of the number of processes on memory pressure:

```
~]$ snmpget localhost \
'NET-SNMP-EXTEND-MIB::nsExtendResult."httpd_pids"' \
UCD-SNMP-MIB::memAvailReal.0
NET-SNMP-EXTEND-MIB::nsExtendResult."httpd_pids" = INTEGER: 8
UCD-SNMP-MIB::memAvailReal.0 = INTEGER: 799664 kB
```

16.7. Additional Resources

To learn more about gathering system information, see the following resources.

16.7.1. Installed Documentation

- ▶ **ps(1)** — The manual page for the **ps** command.
- ▶ **top(1)** — The manual page for the **top** command.
- ▶ **free(1)** — The manual page for the **free** command.
- ▶ **df(1)** — The manual page for the **df** command.
- ▶ **du(1)** — The manual page for the **du** command.
- ▶ **lspci(8)** — The manual page for the **lspci** command.
- ▶ **snmpd(8)** — The manual page for the **snmpd** service.
- ▶ **snmpd.conf(5)** — The manual page for the **/etc/snmp/snmpd.conf** file containing full documentation of available configuration directives.

Chapter 17. OpenLMI

The **Open Linux Management Infrastructure**, commonly abbreviated as **OpenLMI**, is a common infrastructure for the management of Linux systems. It builds on top of existing tools and serves as an abstraction layer in order to hide much of the complexity of the underlying system from system administrators. OpenLMI is distributed with a set of services that can be accessed locally or remotely and provides multiple language bindings, standard APIs, and standard scripting interfaces that can be used to manage and monitor hardware, operating systems, and system services.

17.1. About OpenLMI

OpenLMI is designed to provide a common management interface to production servers running the Red Hat Enterprise Linux system on both physical and virtual machines. It consists of the following three components:

1. *System management agents* — these agents are installed on a managed system and implement an object model that is presented to a standard object broker. The initial agents implemented in OpenLMI include storage configuration and network configuration, but later work will address additional elements of system management. The system management agents are commonly referred to as *Common Information Model providers* or *CIM providers*.
2. A *standard object broker* — the object broker manages system management agents and provides an interface to them. The standard object broker is also known as a *CIM Object Monitor* or *CIMOM*.
3. *Client applications and scripts* — the client applications and scripts call the system management agents through the standard object broker.

The OpenLMI project complements existing management initiatives by providing a low-level interface that can be used by scripts or system management consoles. Interfaces distributed with OpenLMI include C, C++, Python, Java, and an interactive command line client, and all of them offer the same full access to the capabilities implemented in each agent. This ensures that you always have access to exactly the same capabilities no matter which programming interface you decide to use.

17.1.1. Main Features

The following are key benefits of installing and using OpenLMI on your system:

- ▶ OpenLMI provides a standard interface for configuration, management, and monitoring of your local and remote systems.
- ▶ It allows you to configure, manage, and monitor production servers running on both physical and virtual machines.
- ▶ It is distributed with a collection of CIM providers that allow you to configure, manage, and monitor storage devices and complex networks.
- ▶ It allows you to call system management functions from C, C++, Python, and Java programs, and is distributed with a command line interface.
- ▶ It is free software based on open industry standards.

17.1.2. Management Capabilities

Key capabilities of OpenLMI include the management of storage devices, networks, system services, user accounts, hardware and software configuration, power management, and interaction with Active Directory. For a complete list of CIM providers that are distributed with Red Hat Enterprise Linux 7, see [Table 17.1, “Available CIM Providers”](#).

Table 17.1. Available CIM Providers

Package Name	Description
<i>openlmi-account</i>	A CIM provider for managing user accounts.
<i>openlmi-logicalfile</i>	A CIM provider for reading files and directories.
<i>openlmi-networking</i>	A CIM provider for network management.
<i>openlmi-powermanagement</i>	A CIM provider for power management.
<i>openlmi-service</i>	A CIM provider for managing system services.
<i>openlmi-storage</i>	A CIM provider for storage management.
<i>openlmi-fan</i>	A CIM provider for controlling computer fans.
<i>openlmi-hardware</i>	A CIM provider for retrieving hardware information.
<i>openlmi-realmd</i>	A CIM provider for configuring realmd.
<i>openlmi-software</i> [a]	A CIM provider for software management.

[a] In Red Hat Enterprise Linux 7, the OpenLMI Software provider is included as a [Technology Preview](#). This provider is fully functional, but has a known performance scaling issue where listing large numbers of software packages may consume excessive amount of memory and time. To work around this issue, adjust package searches to return as few packages as possible.

17.2. Installing OpenLMI

OpenLMI is distributed as a collection of RPM packages that include the CIMOM, individual CIM providers, and client applications. This allows you distinguish between a managed and client system and install only those components you need.

17.2.1. Installing OpenLMI on a Managed System

A *managed system* is the system you intend to monitor and manage by using the OpenLMI client tools. To install OpenLMI on a managed system, complete the following steps:

1. Install the *tog-pegasus* package by typing the following at a shell prompt as **root**:

```
yum install tog-pegasus
```

This command installs the OpenPegasus CIMOM and all its dependencies to the system and creates a user account for the **pegasus** user.

2. Install required CIM providers by running the following command as **root**:

```
yum install openlmi-{storage,networking,service,account,powermanagement}
```

This command installs the CIM providers for storage, network, service, account, and power management. For a complete list of CIM providers distributed with Red Hat Enterprise Linux 7, see [Table 17.1, “Available CIM Providers”](#).

3. Edit the **/etc/Pegasus/access.conf** configuration file to customize the list of users that are allowed to connect to the OpenPegasus CIMOM. By default, only the **pegasus** user is allowed to access the CIMOM both remotely and locally. To activate this user account, run the following command as **root** to set the user's password:

```
passwd pegasus
```

- Start the OpenPegasus CIMOM by activating the **tog-pegasus.service** unit. To activate the **tog-pegasus.service** unit in the current session, type the following at a shell prompt as **root**:

```
systemctl start tog-pegasus.service
```

To configure the **tog-pegasus.service** unit to start automatically at boot time, type as **root**:

```
systemctl enable tog-pegasus.service
```

- If you intend to interact with the managed system from a remote machine, enable TCP communication on port **5989** (**wbem-https**). To open this port in the current session, run the following command as **root**:

```
firewall-cmd --add-port 5989/tcp
```

To open port **5989** for TCP communication permanently, type as **root**:

```
firewall-cmd --permanent --add-port 5989/tcp
```

You can now connect to the managed system and interact with it by using the OpenLMI client tools as described in [Section 17.4, “Using LMIShell”](#). If you intend to perform OpenLMI operations directly on the managed system, also complete the steps described in [Section 17.2.2, “Installing OpenLMI on a Client System”](#).

17.2.2. Installing OpenLMI on a Client System

A *client system* is the system from which you intend to interact with the managed system. In a typical scenario, the client system and the managed system are installed on two separate machines, but you can also install the client tools on the managed system and interact with it directly.

To install OpenLMI on a client system, complete the following steps:

- Install the *openlmi-tools* package by typing the following at a shell prompt as **root**:

```
yum install openlmi-tools
```

This command installs LMIShell, an interactive client and interpreter for accessing CIM objects provided by OpenPegasus, and all its dependencies to the system.

- Configure SSL certificates for OpenPegasus as described in [Section 17.3, “Configuring SSL Certificates for OpenPegasus”](#).

You can now use the LMIShell client to interact with the managed system as described in [Section 17.4, “Using LMIShell”](#).

17.3. Configuring SSL Certificates for OpenPegasus

OpenLMI uses the Web-Based Enterprise Management (WBEM) protocol that functions over an HTTP transport layer. Standard HTTP Basic authentication is performed in this protocol, which means that the user name and password are transmitted alongside the requests.

Configuring the OpenPegasus CIMOM to use HTTPS for communication is necessary to ensure secure authentication. A Secure Sockets Layer (SSL) or Transport Layer Security (TLS) certificate is required on the managed system to establish an encrypted channel.

There are two ways of managing SSL/TLS certificates on a system:

- ▶ Self-signed certificates require less infrastructure to use, but are more difficult to deploy to clients and manage securely.
- ▶ Authority-signed certificates are easier to deploy to clients once they are set up, but may require a greater initial investment.

When using an authority-signed certificate, it is necessary to configure a trusted certificate authority on the client systems. The authority can then be used for signing all of the managed systems' CIMOM certificates. Certificates can also be part of a certificate chain, so the certificate used for signing the managed systems' certificates may in turn be signed by another, higher authority (such as Verisign, CAcert, RSA and many others).

The default certificate and trust store locations on the file system are listed in [Table 17.2, “Certificate and Trust Store Locations”](#).

Table 17.2. Certificate and Trust Store Locations

Configuration Option	Location	Description
<code>sslCertificateFilePat</code>	<code>/etc/Pegasus/server.pem</code>	Public certificate of the CIMOM.
<code>sslKeyFilePath</code>	<code>/etc/Pegasus/file.pem</code>	Private key known only to the CIMOM.
<code>sslTrustStore</code>	<code>/etc/Pegasus/client.pem</code>	The file or directory providing the list of trusted certificate authorities.



Important

If you modify any of the files mentioned in [Table 17.2, “Certificate and Trust Store Locations”](#), restart the `tog-pegasus` service to make sure it recognizes the new certificates. To restart the service, type the following at a shell prompt as `root`:

```
systemctl restart tog-pegasus.service
```

For more information on how to manage system services in Red Hat Enterprise Linux 7, see [Chapter 6, “Managing Services with systemd”](#).

17.3.1. Managing Self-signed Certificates

A self-signed certificate uses its own private key to sign itself and it is not connected to any chain of trust. On a managed system, if certificates have not been provided by the administrator prior to the first time that the `tog-pegasus` service is started, a set of self-signed certificates will be automatically generated using the system's primary hostname as the certificate subject.



Important

The automatically generated self-signed certificates are valid by default for 10 years, but they have no automatic-renewal capability. Any modification to these certificates will require manually creating new certificates following guidelines provided by the [OpenSSL](#) or [Mozilla NSS](#) documentation on the subject.

To configure client systems to trust the self-signed certificate, complete the following steps:

1. Copy the `/etc/Pegasus/client.pem` certificate from the managed system to the `/etc/pki/ca-trust/source/anchors/` directory on the client system. To do so, type the following at a shell prompt as `root`:

```
scp root@hostname:/etc/Pegasus/client.pem /etc/pki/ca-
trust/source/anchors/pegasus-hostname.pem
```

Replace `hostname` with the host name of the managed system. Note that this command only works if the `sshd` service is running on the managed system and is configured to allow the `root` user to log in to the system over the SSH protocol. For more information on how to install and configure the `sshd` service and use the `scp` command to transfer files over the SSH protocol, see [Chapter 7, OpenSSH](#).

2. Verify the integrity of the certificate on the client system by comparing its checksum with the checksum of the original file. To calculate the checksum of the `/etc/Pegasus/client.pem` file on the managed system, run the following command as `root` on that system:

```
sha1sum /etc/Pegasus/client.pem
```

To calculate the checksum of the `/etc/pki/ca-trust/source/anchors/pegasus-hostname.pem` file on the client system, run the following command on this system:

```
sha1sum /etc/pki/ca-trust/source/anchors/pegasus-hostname.pem
```

Replace `hostname` with the host name of the managed system.

3. Update the trust store by running the following command as `root`:

```
update-ca-trust extract
```

17.3.2. Managing Authority-signed Certificates with Identity Management (Recommended)

The Identity Management feature of Red Hat Enterprise Linux provides a domain controller which simplifies the management of SSL certificates within systems joined to the domain. Among others, the Identity Management server provides an embedded Certificate Authority. See the [Red Hat Enterprise Linux 7 Linux Domain Identity, Authentication, and Policy Guide](#) or the FreeIPA documentation for information on how to join the client and managed systems to the domain.

It is necessary to register the managed system to Identity Management; for client systems the registration is optional.

The following steps are required on the managed system:

1. Install the *ipa-client* package and register the system to Identity Management as described in the [Red Hat Enterprise Linux 7 Linux Domain Identity, Authentication, and Policy Guide](#).
2. Copy the Identity Management signing certificate to the trusted store by typing the following command as **root**:

```
cp /etc/ipa/ca.crt /etc/pki/ca-trust/source/anchors/ipa.crt
```

3. Update the trust store by running the following command as **root**:

```
update-ca-trust extract
```

4. Register Pegasus as a service in the Identity Management domain by running the following command as a privileged domain user:

```
ipa service-add CIMOM/hostname
```

Replace *hostname* with the host name of the managed system.

This command can be run from any system in the Identity Management domain that has the *ipa-admintools* package installed. It creates a service entry in Identity Management that can be used to generate signed SSL certificates.

5. Back up the PEM files located in the */etc/Pegasus/* directory (recommended).
6. Retrieve the signed certificate by running the following command as **root**:

```
ipa-getcert request -f /etc/Pegasus/server.pem -k /etc/Pegasus/file.pem -N  
CN=hostname -K CIMOM/hostname
```

Replace *hostname* with the host name of the managed system.

The certificate and key files are now kept in proper locations. The **certmonger** daemon installed on the managed system by the **ipa-client-install** script ensures that the certificate is kept up-to-date and renewed as necessary.

For more information, see the [Red Hat Enterprise Linux 7 Linux Domain Identity, Authentication, and Policy Guide](#).

To register the client system and update the trust store, follow the steps below.

1. Install the *ipa-client* package and register the system to Identity Management as described in the [Red Hat Enterprise Linux 7 Linux Domain Identity, Authentication, and Policy Guide](#).
2. Copy the Identity Management signing certificate to the trusted store by typing the following command as **root**:

```
cp /etc/ipa/ca.crt /etc/pki/ca-trust/source/anchors/ipa.crt
```

3. Update the trust store by running the following command as **root**:

```
update-ca-trust extract
```

If the client system is not meant to be registered in Identity Management, complete the following steps to update the trust store.

1. Copy the `/etc/ipa/ca.crt` file securely from any other system joined to the same Identity Management domain to the trusted store `/etc/pki/ca-trust/source/anchors/` directory as **root**.
2. Update the trust store by running the following command as **root**:

```
update-ca-trust extract
```

17.3.3. Managing Authority-signed Certificates Manually

Managing authority-signed certificates with other mechanisms than Identity Management requires more manual configuration.

It is necessary to ensure that all of the clients trust the certificate of the authority that will be signing the managed system certificates:

- ▶ If a certificate authority is trusted by default, it is not necessary to perform any particular steps to accomplish this.
 - ▶ If the certificate authority is not trusted by default, the certificate has to be imported on the client and managed systems.
- ▶ Copy the certificate to the trusted store by typing the following command as **root**:

```
cp /path/to/ca.crt /etc/pki/ca-trust/source/anchors/ca.crt
```

- ▶ Update the trust store by running the following command as **root**:

```
update-ca-trust extract
```

On the managed system, complete the following steps:

1. Create a new SSL configuration file located at `/etc/Pegasus/ssl.cnf` to store information about the certificate. The contents of this file must be similar to the following example:

```
[ req ]
distinguished_name      = req_distinguished_name
prompt                  = no
[ req_distinguished_name ]
C                       = US
ST                      = Massachusetts
L                       = Westford
O                       = Fedora
OU                      = Fedora OpenLMI
CN                      = hostname
```

Replace *hostname* with the fully qualified domain name of the managed system.

2. Generate a private key on the managed system by using the following command as **root**:

```
openssl genrsa -out /etc/Pegasus/file.pem 1024
```

3. Generate a certificate signing request (CSR) by running this command as **root**:

```
openssl req -config /etc/Pegasus/ssl.cnf -new -key /etc/Pegasus/file.pem -out /etc/Pegasus/server.csr
```

4. Send the `/etc/Pegasus/server.csr` file to the certificate authority for signing. The detailed procedure of submitting the file depends on the particular certificate authority.
5. When the signed certificate is received from the certificate authority, save it as `/etc/Pegasus/server.pem`.
6. Copy the certificate of the trusted authority to the Pegasus trust store to make sure that Pegasus is capable of trusting its own certificate by running as `root`:

```
cp /path/to/ca.crt /etc/Pegasus/client.pem
```

After accomplishing all the described steps, the clients that trust the signing authority are able to successfully communicate with the managed server's CIMOM.



Important

Unlike the Identity Management solution, if the certificate expires and needs to be renewed, all of the described manual steps have to be carried out again. It is recommended to renew the certificates before they expire.

17.4. Using LMIShell

LMIShell is an interactive client and non-interactive interpreter that can be used to access CIM objects provided by the OpenPegasus CIMOM. It is based on the Python interpreter, but also implements additional functions and classes for interacting with CIM objects.

17.4.1. Starting, Using, and Exiting LMIShell

Similarly to the Python interpreter, you can use LMIShell either as an interactive client, or as a non-interactive interpreter for LMIShell scripts.

Starting LMIShell in Interactive Mode

To start the LMIShell interpreter in interactive mode, run the `lmishell` command with no additional arguments:

```
lmishell
```

By default, when LMIShell attempts to establish a connection with a CIMOM, it validates the server-side certificate against the Certification Authorities trust store. To disable this validation, run the `lmishell` command with the `--noverify` or `-n` command line option:

```
lmishell --noverify
```

Using Tab Completion

When running in interactive mode, the LMIShell interpreter allows you press the **Tab** key to complete basic programming structures and CIM objects, including namespaces, classes, methods, and object properties.

Browsing History

By default, LMIShell stores all commands you type at the interactive prompt in the `~/.lmishell_history` file. This allows you to browse the command history and re-use already entered lines in interactive mode without the need to type them at the prompt again. To move backward in the command history, press the **Up Arrow** key or the **Ctrl+p** key combination. To move forward in the command history, press the **Down Arrow** key or the **Ctrl+n** key combination.

LMIShell also supports an incremental reverse search. To look for a particular line in the command history, press **Ctrl+r** and start typing any part of the command. For example:

```
> (reverse-i-search)`connect': c = connect("server.example.com", "pegasus")
```

To clear the command history, use the `clear_history()` function as follows:

```
clear_history()
```

You can configure the number of lines that are stored in the command history by changing the value of the `history_length` option in the `~/.lmishellrc` configuration file. In addition, you can change the location of the history file by changing the value of the `history_file` option in this configuration file. For example, to set the location of the history file to `~/.lmishell_history` and configure LMIShell to store the maximum of **1000** lines in it, add the following lines to the `~/.lmishellrc` file:

```
history_file = "~/.lmishell_history"
history_length = 1000
```

Handling Exceptions

By default, the LMIShell interpreter handles all exceptions and uses return values. To disable this behavior in order to handle all exceptions in the code, use the `use_exceptions()` function as follows:

```
use_exceptions()
```

To re-enable the automatic exception handling, use:

```
use_exception(False)
```

You can permanently disable the exception handling by changing the value of the `use_exceptions` option in the `~/.lmishellrc` configuration file to **True**:

```
use_exceptions = True
```

Configuring a Temporary Cache

With the default configuration, LMIShell connection objects use a temporary cache for storing CIM class names and CIM classes in order to reduce network communication. To clear this temporary cache, use the `clear_cache()` method as follows:

```
object_name.clear_cache()
```

Replace `object_name` with the name of a connection object.

To disable the temporary cache for a particular connection object, use the `use_cache()` method as follows:

```
object_name.use_cache(False)
```

To enable it again, use:

```
object_name.use_cache(True)
```

You can permanently disable the temporary cache for connection objects by changing the value of the **use_cache** option in the `~/.lmishellrc` configuration file to **False**:

```
use_cache = False
```

Exiting LMIShell

To terminate the LMIShell interpreter and return to the shell prompt, press the **Ctrl+d** key combination or issue the **quit()** function as follows:

```
> quit()  
~]$
```

Running an LMIShell Script

To run an LMIShell script, run the **lmishell** command as follows:

```
lmishell file_name
```

Replace *file_name* with the name of the script. To inspect an LMIShell script after its execution, also specify the **--interact** or **-i** command line option:

```
lmishell --interact file_name
```

The preferred file extension of LMIShell scripts is **.lmi**.

17.4.2. Connecting to a CIMOM

LMIShell allows you to connect to a CIMOM that is running either locally on the same system, or on a remote machine accessible over the network.

Connecting to a Remote CIMOM

To access CIM objects provided by a remote CIMOM, create a connection object by using the **connect()** function as follows:

```
connect(host_name, user_name[, password])
```

Replace *host_name* with the host name of the managed system, *user_name* with the name of a user that is allowed to connect to the OpenPegasus CIMOM running on that system, and *password* with the user's password. If the password is omitted, LMIShell prompts the user to enter it. The function returns an **LMIConnection** object.

Example 17.1. Connecting to a Remote CIMOM

To connect to the OpenPegasus CIMOM running on `server.example.com` as user `pegasus`, type the following at the interactive prompt:

```
> c = connect("server.example.com", "pegasus")
password:
>
```

Connecting to a Local CIMOM

LMIShell allows you to connect to a local CIMOM by using a Unix socket. For this type of connection, you must run the LMIShell interpreter as the `root` user and the `/var/run/tog-pegasus/cimxml.socket` socket must exist.

To access CIM objects provided by a local CIMOM, create a connection object by using the `connect()` function as follows:

```
connect(host_name)
```

Replace `host_name` with `localhost`, `127.0.0.1`, or `::1`. The function returns an `LMIClnection` object or `None`.

Example 17.2. Connecting to a Local CIMOM

To connect to the OpenPegasus CIMOM running on `localhost` as the `root` user, type the following at the interactive prompt:

```
> c = connect("localhost")
>
```

Verifying a Connection to a CIMOM

The `connect()` function returns either an `LMIClnection` object, or `None` if the connection could not be established. In addition, when the `connect()` function fails to establish a connection, it prints an error message to standard error output.

To verify that a connection to a CIMOM has been established successfully, use the `isinstance()` function as follows:

```
isinstance(object_name, LMIClnection)
```

Replace `object_name` with the name of the connection object. This function returns `True` if `object_name` is an `LMIClnection` object, or `False` otherwise.

Example 17.3. Verifying a Connection to a CIMOM

To verify that the `c` variable created in [Example 17.1, “Connecting to a Remote CIMOM”](#) contains an `LMIClnection` object, type the following at the interactive prompt:

```
> isinstance(c, LMIClnection)
True
>
```

Alternatively, you can verify that `c` is not `None`:

```
> c is None
False
>
```

17.4.3. Working with Namespaces

LMIShell namespaces provide a natural means of organizing available classes and serve as a hierachic access point to other namespaces and classes. The `root` namespace is the first entry point of a connection object.

Listing Available Namespaces

To list all available namespaces, use the `print_namespaces()` method as follows:

```
object_name.print_namespaces()
```

Replace `object_name` with the name of the object to inspect. This method prints available namespaces to standard output.

To get a list of available namespaces, access the object attribute `namespaces`:

```
object_name.namespaces
```

This returns a list of strings.

Example 17.4. Listing Available Namespaces

To inspect the `root` namespace object of the `c` connection object created in [Example 17.1, “Connecting to a Remote CIMOM”](#) and list all available namespaces, type the following at the interactive prompt:

```
> c.root.print_namespaces()
cimv2
interop
PG_InterOp
PG_Internal
>
```

To assign a list of these namespaces to a variable named `root_namespaces`, type:

```
> root_namespaces = c.root.namespaces
>
```

Accessing Namespace Objects

To access a particular namespace object, use the following syntax:

```
object_name.namespace_name
```

Replace *object_name* with the name of the object to inspect and *namespace_name* with the name of the namespace to access. This returns an **LMINamespace** object.

Example 17.5. Accessing Namespace Objects

To access the **cimv2** namespace of the **c** connection object created in [Example 17.1, “Connecting to a Remote CIMOM”](#) and assign it to a variable named **ns**, type the following at the interactive prompt:

```
> ns = c.root.cimv2
>
```

17.4.4. Working with Classes

LMIShell classes represent classes provided by a CIMOM. You can access and list their properties, methods, instances, instance names, and ValueMap properties, print their documentation strings, and create new instances and instance names.

Listing Available Classes

To list all available classes in a particular namespace, use the **print_classes()** method as follows:

```
namespace_object.print_classes()
```

Replace *namespace_object* with the namespace object to inspect. This method prints available classes to standard output.

To get a list of available classes, use the **classes()** method:

```
namespace_object.classes()
```

This method returns a list of strings.

Example 17.6. Listing Available Classes

To inspect the `ns` namespace object created in [Example 17.5, “Accessing Namespace Objects”](#) and list all available classes, type the following at the interactive prompt:

```
> ns.print_classes()
CIM_CollectionInSystem
CIM_ConcreteIdentity
CIM_ControlledBy
CIM_DeviceSAPIImplementation
CIM_MemberOfStatusCollection
...
>
```

To assign a list of these classes to a variable named `cimv2_classes`, type:

```
> cimv2_classes = ns.classes()
>
```

Accessing Class Objects

To access a particular class object that is provided by the CIMOM, use the following syntax:

```
namespace_object.class_name
```

Replace `namespace_object` with the name of the namespace object to inspect and `class_name` with the name of the class to access.

Example 17.7. Accessing Class Objects

To access the `LMI_IPNetworkConnection` class of the `ns` namespace object created in [Example 17.5, “Accessing Namespace Objects”](#) and assign it to a variable named `cls`, type the following at the interactive prompt:

```
> cls = ns.LMI_IPNetworkConnection
>
```

Examining Class Objects

All class objects store information about their name and the namespace they belong to, as well as detailed class documentation. To get the name of a particular class object, use the following syntax:

```
class_object.classname
```

Replace `class_object` with the name of the class object to inspect. This returns a string representation of the object name.

To get information about the namespace a class object belongs to, use:

```
class_object.namespace
```

This returns a string representation of the namespace.

To display detailed class documentation, use the **doc()** method as follows:

```
class_object.doc()
```

Example 17.8. Examining Class Objects

To inspect the **cls** class object created in [Example 17.7, “Accessing Class Objects”](#) and display its name and corresponding namespace, type the following at the interactive prompt:

```
> cls.classname
'LMI_IPNetworkConnection'
> cls.namespace
'root/cimv2'
>
```

To access class documentation, type:

```
> cls.doc()
Class: LMI_IPNetworkConnection
    SuperClass: CIM_IPNetworkConnection
    [qualifier] string UMLPackagePath: 'CIM::Network::IP'

    [qualifier] string Version: '0.1.0'
...
```

Listing Available Methods

To list all available methods of a particular class object, use the **print_methods()** method as follows:

```
class_object.print_methods()
```

Replace *class_object* with the name of the class object to inspect. This method prints available methods to standard output.

To get a list of available methods, use the **methods()** method:

```
class_object.methods()
```

This method returns a list of strings.

Example 17.9. Listing Available Methods

To inspect the **cls** class object created in [Example 17.7, “Accessing Class Objects”](#) and list all available methods, type the following at the interactive prompt:

```
> cls.print_methods()
RequestStateChange
>
```

To assign a list of these methods to a variable named **service_methods**, type:

```
> service_methods = cls.methods()
>
```

Listing Available Properties

To list all available properties of a particular class object, use the **print_properties()** method as follows:

```
class_object.print_properties()
```

Replace *class_object* with the name of the class object to inspect. This method prints available properties to standard output.

To get a list of available properties, use the **properties()** method:

```
class_object.properties()
```

This method returns a list of strings.

Example 17.10. Listing Available Properties

To inspect the **cls** class object created in [Example 17.7, “Accessing Class Objects”](#) and list all available properties, type the following at the interactive prompt:

```
> cls.print_properties()
RequestedState
HealthState
StatusDescriptions
TransitioningToState
Generation
...
>
```

To assign a list of these classes to a variable named **service_properties**, type:

```
> service_properties = cls.properties()
>
```

Listing and Viewing ValueMap Properties

CIM classes may contain *ValueMap properties* in their Managed Object Format (MOF) definition. ValueMap properties contain constant values, which may be useful when calling methods or checking returned values.

To list all available ValueMap properties of a particular class object, use the **print_valuemap_properties()** method as follows:

```
class_object.print_valuemap_properties()
```

Replace *class_object* with the name of the class object to inspect. This method prints available ValueMap properties to standard output.

To get a list of available ValueMap properties, use the **valuemap_properties()** method:

```
class_object.valuemap_properties()
```

This method returns a list of strings.

Example 17.11. Listing ValueMap Properties

To inspect the `cls` class object created in [Example 17.7, “Accessing Class Objects”](#) and list all available ValueMap properties, type the following at the interactive prompt:

```
> cls.print_valuemap_properties()
RequestedState
HealthState
TransitioningToState
DetailedStatus
OperationalStatus
...
>
```

To assign a list of these ValueMap properties to a variable named `service_valuemap_properties`, type:

```
> service_valuemap_properties = cls.valuemap_properties()
>
```

To access a particular ValueMap property, use the following syntax:

```
class_object.valuemap_propertyValues
```

Replace `valuemap_property` with the name of the ValueMap property to access.

To list all available constant values, use the `print_values()` method as follows:

```
class_object.valuemap_propertyValues.print_values()
```

This method prints available named constant values to standard output. You can also get a list of available constant values by using the `values()` method:

```
class_object.valuemap_propertyValues.values()
```

This method returns a list of strings.

Example 17.12. Accessing ValueMap Properties

[Example 17.11, “Listing ValueMap Properties”](#) mentions a ValueMap property named **RequestedState**. To inspect this property and list available constant values, type the following at the interactive prompt:

```
> cls.RequestedStateValues.print_values()
Reset
NoChange
NotApplicable
Quiesce
Unknown
...
>
```

To assign a list of these constant values to a variable named **requested_state_values**, type:

```
> requested_state_values = cls.RequestedStateValues.values()
>
```

To access a particular constant value, use the following syntax:

```
class_object.valuemap_propertyValues.constant_value_name
```

Replace *constant_value_name* with the name of the constant value. Alternatively, you can use the **value()** method as follows:

```
class_object.valuemap_propertyValues.value("constant_value_name")
```

To determine the name of a particular constant value, use the **value_name()** method:

```
class_object.valuemap_propertyValues.value_name("constant_value")
```

This method returns a string.

Example 17.13. Accessing Constant Values

[Example 17.12, “Accessing ValueMap Properties”](#) shows that the **RequestedState** property provides a constant value named **Reset**. To access this named constant value, type the following at the interactive prompt:

```
> cls.RequestedStateValues.Reset
11
> cls.RequestedStateValues.value("Reset")
11
>
```

To determine the name of this constant value, type:

```
> cls.RequestedStateValues.value_name(11)
u'Reset'
>
```

Fetching a CIMClass Object

Many class methods do not require access to a **CIMClass** object, which is why LMIShell only fetches this object from the CIMOM when a called method actually needs it. To fetch the **CIMClass** object manually, use the **fetch()** method as follows:

```
class_object.fetch()
```

Replace *class_object* with the name of the class object. Note that methods that require access to a **CIMClass** object fetch it automatically.

17.4.5. Working with Instances

LMIShell instances represent instances provided by a CIMOM. You can get and set their properties, list and call their methods, print their documentation strings, get a list of associated or association objects, push modified objects to the CIMOM, and delete individual instances from the CIMOM.

Accessing Instances

To get a list of all available instances of a particular class object, use the **instances()** method as follows:

```
class_object.instances()
```

Replace *class_object* with the name of the class object to inspect. This method returns a list of **LMIInstance** objects.

To access the first instance of a class object, use the **first_instance()** method:

```
class_object.first_instance()
```

This method returns an **LMIInstance** object.

In addition to listing all instances or returning the first one, both **instances()** and **first_instance()** support an optional argument to allow you to filter the results:

```
class_object.instances(criteria)
```

```
class_object.first_instance(criteria)
```

Replace *criteria* with a dictionary consisting of key-value pairs, where keys represent instance properties and values represent required values of these properties.

Example 17.14. Accessing Instances

To find the first instance of the **cls** class object created in [Example 17.7, “Accessing Class Objects”](#) that has the **ElementName** property equal to **eth0** and assign it to a variable named **device**, type the following at the interactive prompt:

```
> device = cls.first_instance({"ElementName": "eth0"})
```

Examining Instances

All instance objects store information about their class name and the namespace they belong to, as well as detailed documentation about their properties and values. In addition, instance objects allow you to retrieve a unique identification object.

To get the class name of a particular instance object, use the following syntax:

```
instance_object.classname
```

Replace *instance_object* with the name of the instance object to inspect. This returns a string representation of the class name.

To get information about the namespace an instance object belongs to, use:

```
instance_object.namespace
```

This returns a string representation of the namespace.

To retrieve a unique identification object for an instance object, use:

```
instance_object.path
```

This returns an **LMIInstanceName** object.

Finally, to display detailed documentation, use the **doc()** method as follows:

```
instance_object.doc()
```

Example 17.15. Examining Instances

To inspect the **device** instance object created in [Example 17.14, “Accessing Instances”](#) and display its class name and the corresponding namespace, type the following at the interactive prompt:

```
> device.classname
u'LMI_IPNetworkConnection'
> device.namespace
'root/cimv2'
>
```

To access instance object documentation, type:

```
> device.doc()
Instance of LMI_IPNetworkConnection
[property] uint16 RequestedState = '12'

[property] uint16 HealthState

[property array] string [] StatusDescriptions
...
```

Creating New Instances

Certain CIM providers allow you to create new instances of specific classes objects. To create a new instance of a class object, use the **create_instance()** method as follows:

```
class_object.create_instance(properties)
```

Replace *class_object* with the name of the class object and *properties* with a dictionary that consists of key-value pairs, where keys represent instance properties and values represent property values. This method returns an **LMIInstance** object.

Example 17.16. Creating New Instances

The **LMI_Group** class represents system groups and the **LMI_Account** class represents user accounts on the managed system. To use the **ns** namespace object created in [Example 17.5, “Accessing Namespace Objects”](#), create instances of these two classes for the system group named **pegasus** and the user named **lmishell-user**, and assign them to variables named **group** and **user**, type the following at the interactive prompt:

```
> group = ns.LMI_Group.first_instance({"Name" : "pegasus"})
> user = ns.LMI_Account.first_instance({"Name" : "lmishell-user"})
>
```

To get an instance of the **LMI_Identity** class for the **lmishell-user** user, type:

```
> identity = user.first_associator(ResultClass="LMI_Identity")
>
```

The **LMI_MemberOfGroup** class represents system group membership. To use the **LMI_MemberOfGroup** class to add the **lmishell-user** to the **pegasus** group, create a new instance of this class as follows:

```
> ns.LMI_MemberOfGroup.create_instance({
...     "Member" : identity.path,
...     "Collection" : group.path})
LMIInstance(classname="LMI_MemberOfGroup", ...)
>
```

Deleting Individual Instances

To delete a particular instance from the CIMOM, use the **delete()** method as follows:

```
instance_object.delete()
```

Replace *instance_object* with the name of the instance object to delete. This method returns a boolean. Note that after deleting an instance, its properties and methods become inaccessible.

Example 17.17. Deleting Individual Instances

The **LMI_Account** class represents user accounts on the managed system. To use the **ns** namespace object created in [Example 17.5, “Accessing Namespace Objects”](#), create an instance of the **LMI_Account** class for the user named **lmishell-user**, and assign it to a variable named **user**, type the following at the interactive prompt:

```
> user = ns.LMI_Account.first_instance({"Name" : "lmishell-user"})
>
```

To delete this instance and remove the **lmishell-user** from the system, type:

```
> user.delete()
True
>
```

Listing and Accessing Available Properties

To list all available properties of a particular instance object, use the **print_properties()** method as follows:

```
instance_object.print_properties()
```

Replace *instance_object* with the name of the instance object to inspect. This method prints available properties to standard output.

To get a list of available properties, use the **properties()** method:

```
instance_object.properties()
```

This method returns a list of strings.

Example 17.18. Listing Available Properties

To inspect the **device** instance object created in [Example 17.14, “Accessing Instances”](#) and list all available properties, type the following at the interactive prompt:

```
> device.print_properties()
RequestedState
HealthState
StatusDescriptions
TransitioningToState
Generation
...
>
```

To assign a list of these properties to a variable named **device_properties**, type:

```
> device_properties = device.properties()
>
```

To get the current value of a particular property, use the following syntax:

```
instance_object.property_name
```

Replace *property_name* with the name of the property to access.

To modify the value of a particular property, assign a value to it as follows:

```
instance_object.property_name = value
```

Replace *value* with the new value of the property. Note that in order to propagate the change to the CIMOM, you must also execute the **push()** method:

```
instance_object.push()
```

This method returns a three-item tuple consisting of a return value, return value parameters, and an error string.

Example 17.19. Accessing Individual Properties

To inspect the **device** instance object created in [Example 17.14, “Accessing Instances”](#) and display the value of the property named **SystemName**, type the following at the interactive prompt:

```
> device.SystemName  
u'server.example.com'  
>
```

Listing and Using Available Methods

To list all available methods of a particular instance object, use the **print_methods()** method as follows:

```
instance_object.print_methods()
```

Replace *instance_object* with the name of the instance object to inspect. This method prints available methods to standard output.

To get a list of available methods, use the **method()** method:

```
instance_object.methods()
```

This method returns a list of strings.

Example 17.20. Listing Available Methods

To inspect the `device` instance object created in [Example 17.14, “Accessing Instances”](#) and list all available methods, type the following at the interactive prompt:

```
> device.print_methods()  
RequestStateChange  
>
```

To assign a list of these methods to a variable named `network_device_methods`, type:

```
> network_device_methods = device.methods()  
>
```

To call a particular method, use the following syntax:

```
instance_object.method_name(  
    parameter=value,  
    ...)
```

Replace `instance_object` with the name of the instance object to use, `method_name` with the name of the method to call, `parameter` with the name of the parameter to set, and `value` with the value of this parameter. Methods return a three-item tuple consisting of a return value, return value parameters, and an error string.



Important

`LMIInstance` objects do **not** automatically refresh their contents (properties, methods, qualifiers, and so on). To do so, use the `refresh()` method as described below.

Example 17.21. Using Methods

The **PG_ComputerSystem** class represents the system. To create an instance of this class by using the **ns** namespace object created in [Example 17.5, “Accessing Namespace Objects”](#) and assign it to a variable named **sys**, type the following at the interactive prompt:

```
> sys = ns.PG_ComputerSystem.first_instance()
>
```

The **LMI_AccountManagementService** class implements methods that allow you to manage users and groups in the system. To create an instance of this class and assign it to a variable named **acc**, type:

```
> acc = ns.LMI_AccountManagementService.first_instance()
>
```

To create a new user named **lmishell-user** in the system, use the **CreateAccount()** method as follows:

```
> acc.CreateAccount(Name="lmishell-user", System=sys)
LMIReturnValue(rval=0, rparams=NocaseDict({u'Account':
LMIInstanceName(classname="LMI_Account"...), u'Identities':
[LMIInstanceName(classname="LMI_Identity"...),
LMIInstanceName(classname="LMI_Identity"...)]}), errorstr='')
```

LMIShell support synchronous method calls: when you use a synchronous method, LMIShell waits for the corresponding Job object to change its state to “finished” and then returns the return parameters of this job. LMIShell is able to perform a synchronous method call if the given method returns an object of one of the following classes:

- » **LMI_StorageJob**
- » **LMI_SoftwareInstallationJob**
- » **LMI_NetworkJob**

LMIShell first tries to use indications as the waiting method. If it fails, it uses a polling method instead.

To perform a synchronous method call, use the following syntax:

```
instance_object.Syncmethod_name(
    parameter=value,
    ...)
```

Replace *instance_object* with the name of the instance object to use, *method_name* with the name of the method to call, *parameter* with the name of the parameter to set, and *value* with the value of this parameter. All synchronous methods have the **Sync** prefix in their name and return a three-item tuple consisting of the job's return value, job's return value parameters, and job's error string.

You can also force LMIShell to use only polling method. To do so, specify the **PreferPolling** parameter as follows:

```
instance_object.Syncmethod_name(
    PreferPolling=True
    parameter=value,
    ...)
```

Listing and Viewing ValueMap Parameters

CIM methods may contain *ValueMap parameters* in their Managed Object Format (MOF) definition. ValueMap parameters contain constant values.

To list all available ValueMap parameters of a particular method, use the `print_valuemap_parameters()` method as follows:

```
instance_object.method_name.print_valuemap_parameters()
```

Replace `instance_object` with the name of the instance object and `method_name` with the name of the method to inspect. This method prints available ValueMap parameters to standard output.

To get a list of available ValueMap parameters, use the `valuemap_parameters()` method:

```
instance_object.method_name.valuemap_parameters()
```

This method returns a list of strings.

Example 17.22. Listing ValueMap Parameters

To inspect the `acc` instance object created in [Example 17.21, “Using Methods”](#) and list all available ValueMap parameters of the `CreateAccount()` method, type the following at the interactive prompt:

```
> acc.CreateAccount.print_valuemap_parameters()
CreateAccount
>
```

To assign a list of these ValueMap parameters to a variable named `create_account_parameters`, type:

```
> create_account_parameters = acc.CreateAccount.valuemap_parameters()
>
```

To access a particular ValueMap parameter, use the following syntax:

```
instance_object.method_name.valuemap_parameterValues
```

Replace `valuemap_parameter` with the name of the ValueMap parameter to access.

To list all available constant values, use the `print_values()` method as follows:

```
instance_object.method_name.valuemap_parameterValues.print_values()
```

This method prints available named constant values to standard output. You can also get a list of available constant values by using the `values()` method:

```
instance_object.method_name.valuemap_parameterValues.values()
```

This method returns a list of strings.

Example 17.23. Accessing ValueMap Parameters

[Example 17.22, “Listing ValueMap Parameters”](#) mentions a ValueMap parameter named **CreateAccount**. To inspect this parameter and list available constant values, type the following at the interactive prompt:

```
> acc.CreateAccount.CreateAccountValues.print_values()
Operationunsupported
Failed
Unabletosetpasswordusercreated
Unabletocreathomedirectoryusercreatedandpasswordset
Operationcompletedsuccessfully
>
```

To assign a list of these constant values to a variable named **create_account_values**, type:

```
> create_account_values = acc.CreateAccount.CreateAccountValues.values()
>
```

To access a particular constant value, use the following syntax:

```
instance_object.method_name.valuemap_parameterValues.constant_value_name
```

Replace *constant_value_name* with the name of the constant value. Alternatively, you can use the **value()** method as follows:

```
instance_object.method_name.valuemap_parameterValues.value("constant_value_name")
```

To determine the name of a particular constant value, use the **value_name()** method:

```
instance_object.method_name.valuemap_parameterValues.value_name("constant_value")
```

This method returns a string.

Example 17.24. Accessing Constant Values

[Example 17.23, “Accessing ValueMap Parameters”](#) shows that the **CreateAccount** ValueMap parameter provides a constant value named **Failed**. To access this named constant value, type the following at the interactive prompt:

```
> acc.CreateAccount.CreateAccountValues.Failed
2
> acc.CreateAccount.CreateAccountValues.value("Failed")
2
>
```

To determine the name of this constant value, type:

```
> acc.CreateAccount.CreateAccountValues.value_name(2)
u'Failed'
>
```

Refreshing Instance Objects

Local objects used by LMIShell, which represent CIM objects at CIMOM side, can get outdated, if such objects change while working with LMIShell's ones. To update the properties and methods of a particular instance object, use the `refresh()` method as follows:

```
instance_object.refresh()
```

Replace `instance_object` with the name of the object to refresh. This method returns a three-item tuple consisting of a return value, return value parameter, and an error string.

Example 17.25. Refreshing Instance Objects

To update the properties and methods of the `device` instance object created in [Example 17.14, “Accessing Instances”](#), type the following at the interactive prompt:

```
> device.refresh()
LMIReturnValue(rval=True, rparams=NocaseDict({}), errorstr='')
```

Displaying MOF Representation

To display the Managed Object Format (MOF) representation of an instance object, use the `tomof()` method as follows:

```
instance_object.tomof()
```

Replace `instance_object` with the name of the instance object to inspect. This method prints the MOF representation of the object to standard output.

Example 17.26. Displaying MOF Representation

To display the MOF representation of the `device` instance object created in [Example 17.14, “Accessing Instances”](#), type the following at the interactive prompt:

```
> device.tomof()
instance of LMI_IPNetworkConnection {
    RequestedState = 12;
    HealthState = NULL;
    StatusDescriptions = NULL;
    TransitioningToState = 12;
    ...
}
```

17.4.6. Working with Instance Names

LMIShell instance names are objects that hold a set of primary keys and their values. This type of an object exactly identifies an instance.

Accessing Instance Names

`CIMInstance` objects are identified by `CIMInstanceName` objects. To get a list of all available instance name objects, use the `instance_names()` method as follows:

```
class_object.instance_names()
```

Replace `class_object` with the name of the class object to inspect. This method returns a list of `LMIInstanceName` objects.

To access the first instance name object of a class object, use the `first_instance_name()` method:

```
class_object.first_instance_name()
```

This method returns an `LMIInstanceName` object.

In addition to listing all instance name objects or returning the first one, both `instance_names()` and `first_instance_name()` support an optional argument to allow you to filter the results:

```
class_object.instance_names(criteria)
```

```
class_object.first_instance_name(criteria)
```

Replace `criteria` with a dictionary consisting of key-value pairs, where keys represent key properties and values represent required values of these key properties.

Example 17.27. Accessing Instance Names

To find the first instance name of the `cls` class object created in [Example 17.7. “Accessing Class Objects”](#) that has the `Name` key property equal to `eth0` and assign it to a variable named `device_name`, type the following at the interactive prompt:

```
> device_name = cls.first_instance_name({"Name": "eth0"})  
>
```

Examining Instance Names

All instance name objects store information about their class name and the namespace they belong to.

To get the class name of a particular instance name object, use the following syntax:

```
instance_name_object.classname
```

Replace `instance_name_object` with the name of the instance name object to inspect. This returns a string representation of the class name.

To get information about the namespace an instance name object belongs to, use:

```
instance_name_object.namespace
```

This returns a string representation of the namespace.

Example 17.28. Examining Instance Names

To inspect the `device_name` instance name object created in [Example 17.27, “Accessing Instance Names”](#) and display its class name and the corresponding namespace, type the following at the interactive prompt:

```
> device_name.classname
u'LMI_IPNetworkConnection'
> device_name.namespace
'root/cimv2'
>
```

Creating New Instance Names

LMIShell allows you to create a new wrapped `CIMInstanceName` object if you know all primary keys of a remote object. This instance name object can then be used to retrieve the whole instance object.

To create a new instance name of a class object, use the `new_instance_name()` method as follows:

```
class_object.new_instance_name(key_properties)
```

Replace `class_object` with the name of the class object and `key_properties` with a dictionary that consists of key-value pairs, where keys represent key properties and values represent key property values. This method returns an `LMIInstanceName` object.

Example 17.29. Creating New Instance Names

The `LMI_Account` class represents user accounts on the managed system. To use the `ns` namespace object created in [Example 17.5, “Accessing Namespace Objects”](#) and create a new instance name of the `LMI_Account` class representing the `lmishell-user` user on the managed system, type the following at the interactive prompt:

```
> instance_name = ns.LMI_Account.new_instance_name({
...     "CreationClassName" : "LMI_Account",
...     "Name" : "lmishell-user",
...     "SystemCreationClassName" : "PG_ComputerSystem",
...     "SystemName" : "server"})
>
```

Listing and Accessing Key Properties

To list all available key properties of a particular instance name object, use the `print_key_properties()` method as follows:

```
instance_name_object.print_key_properties()
```

Replace `instance_name_object` with the name of the instance name object to inspect. This method prints available key properties to standard output.

To get a list of available key properties, use the `key_properties()` method:

```
instance_name_object.key_properties()
```

This method returns a list of strings.

Example 17.30. Listing Available Key Properties

To inspect the `device_name` instance name object created in [Example 17.27, “Accessing Instance Names”](#) and list all available key properties, type the following at the interactive prompt:

```
> device_name.print_key_properties()
CreationClassName
SystemName
Name
SystemCreationClassName
>
```

To assign a list of these key properties to a variable named `device_name_properties`, type:

```
> device_name_properties = device_name.key_properties()
>
```

To get the current value of a particular key property, use the following syntax:

```
instance_name_object.key_property_name
```

Replace `key_property_name` with the name of the key property to access.

Example 17.31. Accessing Individual Key Properties

To inspect the `device_name` instance name object created in [Example 17.27, “Accessing Instance Names”](#) and display the value of the key property named `SystemName`, type the following at the interactive prompt:

```
> device_name.SystemName
u'server.example.com'
>
```

Converting Instance Names to Instances

Each instance name can be converted to an instance. To do so, use the `to_instance()` method as follows:

```
instance_name_object.to_instance()
```

Replace `instance_name_object` with the name of the instance name object to convert. This method returns an `LMIInstance` object.

Example 17.32. Converting Instance Names to Instances

To convert the `device_name` instance name object created in [Example 17.27, “Accessing Instance Names”](#) to an instance object and assign it to a variable named `device`, type the following at the interactive prompt:

```
> device = device_name.to_instance()
>
```

17.4.7. Working with Associated Objects

The Common Information Model defines an association relationship between managed objects.

Accessing Associated Instances

To get a list of all objects associated with a particular instance object, use the `associators()` method as follows:

```
instance_object.associators(
    AssocClass=class_name,
    ResultClass=class_name,
    ResultRole=role,
    IncludeQualifiers=include_qualifiers,
    IncludeClassOrigin=include_class_origin,
    PropertyList=property_list)
```

To access the first object associated with a particular instance object, use the `first_associator()` method:

```
instance_object.first_associator(
    AssocClass=class_name,
    ResultClass=class_name,
    ResultRole=role,
    IncludeQualifiers=include_qualifiers,
    IncludeClassOrigin=include_class_origin,
    PropertyList=property_list)
```

Replace `instance_object` with the name of the instance object to inspect. You can filter the results by specifying the following parameters:

- ▶ **AssocClass** — Each returned object must be associated with the source object through an instance of this class or one of its subclasses. The default value is **None**.
- ▶ **ResultClass** — Each returned object must be either an instance of this class or one of its subclasses, or it must be this class or one of its subclasses. The default value is **None**.
- ▶ **Role** — Each returned object must be associated with the source object through an association in which the source object plays the specified role. The name of the property in the association class that refers to the source object must match the value of this parameter. The default value is **None**.
- ▶ **ResultRole** — Each returned object must be associated with the source object through an association in which the returned object plays the specified role. The name of the property in the association class that refers to the returned object must match the value of this parameter. The default value is **None**.

The remaining parameters refer to:

- ▶ **IncludeQualifiers** — A boolean indicating whether all qualifiers of each object (including qualifiers on the object and on any returned properties) should be included as QUALIFIER elements in the response. The default value is **False**.
- ▶ **IncludeClassOrigin** — A boolean indicating whether the CLASSORIGIN attribute should be present on all appropriate elements in each returned object. The default value is **False**.
- ▶ **PropertyList** — The members of this list define one or more property names. Returned objects will not include elements for any properties missing from this list. If **PropertyList** is an empty list, no properties are included in returned objects. If it is **None**, no additional filtering is defined. The default value is **None**.

Example 17.33. Accessing Associated Instances

The **LMI_StorageExtent** class represents block devices available in the system. To use the **ns** namespace object created in [Example 17.5, “Accessing Namespace Objects”](#), create an instance of the **LMI_StorageExtent** class for the block device named `/dev/vda`, and assign it to a variable named **vda**, type the following at the interactive prompt:

```
> vda = ns.LMI_StorageExtent.first_instance({
...     "DeviceID" : "/dev/vda"})
>
```

To get a list of all disk partitions on this block device and assign it to a variable named **vda_partitions**, use the **associators()** method as follows:

```
> vda_partitions = vda.associators(ResultClass="LMI_DiskPartition")
>
```

Accessing Associated Instance Names

To get a list of all associated instance names of a particular instance object, use the **associator_names()** method as follows:

```
instance_object.associator_names(
    AssocClass=class_name,
    ResultClass=class_name,
    Role=role,
    ResultRole=role)
```

To access the first associated instance name of a particular instance object, use the **first_associator_name()** method:

```
instance_object.first_associator_name(
    AssocClass=class_object,
    ResultClass=class_object,
    Role=role,
    ResultRole=role)
```

Replace *instance_object* with the name of the instance object to inspect. You can filter the results by specifying the following parameters:

- ▶ **AssocClass** — Each returned name identifies an object that must be associated with the source object through an instance of this class or one of its subclasses. The default value is **None**.
- ▶ **ResultClass** — Each returned name identifies an object that must be either an instance of this class or one of its subclasses, or it must be this class or one of its subclasses. The default value is **None**.
- ▶ **Role** — Each returned name identifies an object that must be associated with the source object through an association in which the source object plays the specified role. The name of the property in the association class that refers to the source object must match the value of this parameter. The default value is **None**.
- ▶ **ResultRole** — Each returned name identifies an object that must be associated with the source object through an association in which the returned named object plays the specified role. The name of the property in the association class that refers to the returned object must match the value of this parameter. The default value is **None**.

Example 17.34. Accessing Associated Instance Names

To use the **vda** instance object created in [Example 17.33, “Accessing Associated Instances”](#), get a list of its associated instance names, and assign it to a variable named **vda_partitions**, type:

```
> vda_partitions = vda.associator_names(ResultClass="LMI_DiskPartition")
>
```

17.4.8. Working with Association Objects

The Common Information Model defines an association relationship between managed objects. Association objects define the relationship between two other objects.

Accessing Association Instances

To get a list of association objects that refer to a particular target object, use the **references()** method as follows:

```
instance_object.references(
    ResultClass=class_name,
    Role=role,
    IncludeQualifiers=include_qualifiers,
    IncludeClassOrigin=include_class_origin,
    PropertyList=property_list)
```

To access the first association object that refers to a particular target object, use the **first_reference()** method:

```
instance_object.first_reference(
...     ResultClass=class_name,
...     Role=role,
...     IncludeQualifiers=include_qualifiers,
...     IncludeClassOrigin=include_class_origin,
...     PropertyList=property_list)
>
```

Replace *instance_object* with the name of the instance object to inspect. You can filter the results by specifying the following parameters:

- ▶ **ResultClass** — Each returned object must be either an instance of this class or one of its subclasses, or it must be this class or one of its subclasses. The default value is **None**.
- ▶ **Role** — Each returned object must refer to the target object through a property with a name that matches the value of this parameter. The default value is **None**.

The remaining parameters refer to:

- ▶ **IncludeQualifiers** — A boolean indicating whether each object (including qualifiers on the object and on any returned properties) should be included as a QUALIFIER element in the response. The default value is **False**.
- ▶ **IncludeClassOrigin** — A boolean indicating whether the CLASSORIGIN attribute should be present on all appropriate elements in each returned object. The default value is **False**.
- ▶ **PropertyList** — The members of this list define one or more property names. Returned objects will not include elements for any properties missing from this list. If **PropertyList** is an empty list, no properties are included in returned objects. If it is **None**, no additional filtering is defined. The default value is **None**.

Example 17.35. Accessing Association Instances

The **LMI_LANEndpoint** class represents a communication endpoint associated with a certain network interface device. To use the **ns** namespace object created in [Example 17.5, “Accessing Namespace Objects”](#), create an instance of the **LMI_LANEndpoint** class for the network interface device named **eth0**, and assign it to a variable named **lan_endpoint**, type the following at the interactive prompt:

```
> lan_endpoint = ns.LMI_LANEndpoint.first_instance({
...     "Name" : "eth0"})
>
```

To access the first association object that refers to an **LMI_BindsToLANEndpoint** object and assign it to a variable named **bind**, type:

```
> bind = lan_endpoint.first_reference(
...     ResultClass="LMI_BindsToLANEndpoint")
>
```

You can now use the **Dependent** property to access the dependent **LMI_IPProtocolEndpoint** class that represents the IP address of the corresponding network interface device:

```
> ip = bind.Dependent.to_instance()
> print ip.IPv4Address
192.168.122.1
>
```

Accessing Association Instance Names

To get a list of association instance names of a particular instance object, use the **reference_names()** method as follows:

```
instance_object.reference_names(
    ResultClass=class_name,
    Role=role)
```

To access the first association instance name of a particular instance object, use the `first_reference_name()` method:

```
instance_object.first_reference_name(
    ResultClass=class_name,
    Role=role)
```

Replace `instance_object` with the name of the instance object to inspect. You can filter the results by specifying the following parameters:

- ▶ **ResultClass** — Each returned object name identifies either an instance of this class or one of its subclasses, or this class or one of its subclasses. The default value is **None**.
- ▶ **Role** — Each returned object identifies an object that refers to the target instance through a property with a name that matches the value of this parameter. The default value is **None**.

Example 17.36. Accessing Association Instance Names

To use the `lan_endpoint` instance object created in [Example 17.35, “Accessing Association Instances”](#), access the first association instance name that refers to an `LMI_BindsToLANEndpoint` object, and assign it to a variable named `bind`, type:

```
> bind = lan_endpoint.first_reference_name(
...     ResultClass="LMI_BindsToLANEndpoint")
```

You can now use the `Dependent` property to access the dependent `LMI_IPProtocolEndpoint` class that represents the IP address of the corresponding network interface device:

```
> ip = bind.Dependent.to_instance()
> print ip.IPv4Address
192.168.122.1
>
```

17.4.9. Working with Indications

Indication is a reaction to a specific event that occurs in response to a particular change in data. LMIShell can subscribe to an indication in order to receive such event responses.

Subscribing to Indications

To subscribe to an indication, use the `subscribe_indication()` method as follows:

```
connection_object.subscribe_indication(
    QueryLanguage="WQL",
    Query='SELECT * FROM CIM_InstModification',
    Name="cpu",
    CreationNamespace="root/interop",
    SubscriptionCreationClassName="CIM_IndicationSubscription",
    FilterCreationClassName="CIM_IndicationFilter",
    FilterSystemCreationClassName="CIM_ComputerSystem",
    FilterSourceNamespace="root/cimv2",
    HandlerCreationClassName="CIM_IndicationHandlerCIMXML",
    HandlerSystemCreationClassName="CIM_ComputerSystem",
    Destination="http://host_name:5988")
```

Alternatively, you can use a shorter version of the method call as follows:

```
connection_object.subscribe_indication(
    Query='SELECT * FROM CIM_InstModification',
    Name="cpu",
    Destination="http://host_name:5988")
```

Replace `connection_object` with a connection object and `host_name` with the host name of the system you want to deliver the indications to.

By default, all subscriptions created by the LMIShell interpreter are automatically deleted when the interpreter terminates. To change this behavior, pass the **Permanent=True** keyword parameter to the `subscribe_indication()` method call. This will prevent LMIShell from deleting the subscription.

Example 17.37. Subscribing to Indications

To use the `c` connection object created in [Example 17.1, “Connecting to a Remote CIMOM”](#) and subscribe to an indication named `cpu`, type the following at the interactive prompt:

```
> c.subscribe_indication(
...     QueryLanguage="WQL",
...     Query='SELECT * FROM CIM_InstModification',
...     Name="cpu",
...     CreationNamespace="root/interop",
...     SubscriptionCreationClassName="CIM_IndicationSubscription",
...     FilterCreationClassName="CIM_IndicationFilter",
...     FilterSystemCreationClassName="CIM_ComputerSystem",
...     FilterSourceNamespace="root/cimv2",
...     HandlerCreationClassName="CIM_IndicationHandlerCIMXML",
...     HandlerSystemCreationClassName="CIM_ComputerSystem",
...     Destination="http://server.example.com:5988")
LMIReturnValue(rval=True, rparams=NocaseDict({}), errorstr='')
```

Listing Subscribed Indications

To list all the subscribed indications, use the `print_subscribed_indications()` method as follows:

```
connection_object.print_subscribed_indications()
```

Replace `connection_object` with the name of the connection object to inspect. This method prints subscribed indications to standard output.

To get a list of subscribed indications, use the `subscribed_indications()` method:

```
connection_object.subscribed_indications()
```

This method returns a list of strings.

Example 17.38. Listing Subscribed Indications

To inspect the `c` connection object created in [Example 17.1, “Connecting to a Remote CIMOM”](#) and list all subscribed indications, type the following at the interactive prompt:

```
> c.print_subscribed_indications()
>
```

To assign a list of these indications to a variable named `indications`, type:

```
> indications = c.subscribed_indications()
>
```

Unsubscribing from Indications

By default, all subscriptions created by the LMIShell interpreter are automatically deleted when the interpreter terminates. To delete an individual subscription sooner, use the `unsubscribe_indication()` method as follows:

```
connection_object.unsubscribe_indication(indication_name)
```

Replace `connection_object` with the name of the connection object and `indication_name` with the name of the indication to delete.

To delete all subscriptions, use the `unsubscribe_all_indications()` method:

```
connection_object.unsubscribe_all_indications()
```

Example 17.39. Unsubscribing from Indications

To use the `c` connection object created in [Example 17.1, “Connecting to a Remote CIMOM”](#) and unsubscribe from the indication created in [Example 17.37, “Subscribing to Indications”](#), type the following at the interactive prompt:

```
> c.unsubscribe_indication('cpu')
LMIReturnValue(rval=True, rparams=NocaseDict({}), errorstr='')
```

Implementing an Indication Handler

The `subscribe_indication()` method allows you to specify the host name of the system you want to deliver the indications to. The following example shows how to implement an indication handler:

```
> def handler(ind, arg1, arg2, **kwargs):
...     exported_objects = ind.exported_objects()
...     do_something_with(exported_objects)
> listener = LmiIndicationListener("0.0.0.0", listening_port)
> listener.add_handler("indication-name-XXXXXXX", handler, arg1, arg2,
**kwargs)
> listener.start()
>
```

The first argument of the handler is an **LmiIndication** object, which contains a list of methods and objects exported by the indication. Other parameters are user specific: those arguments need to be specified when adding a handler to the listener.

In the example above, the **add_handler()** method call uses a special string with eight “X” characters. These characters are replaced with a random string that is generated by listeners in order to avoid a possible handler name collision. To use the random string, start the indication listener first and then subscribe to an indication so that the **Destination** property of the handler object contains the following value: **schema://host_name/random_string**.

Example 17.40. Implementing an Indication Handler

The following script illustrates how to write a handler that monitors a managed system located at **192.168.122.1** and calls the **indication_callback()** function whenever a new user account is created:

```
#!/usr/bin/lmishell

import sys
from time import sleep
from lmi.shell.LMUtil import LMIPassByRef
from lmi.shell.LMIIIndicationListener import LMIIIndicationListener

# These are passed by reference to indication_callback
var1 = LMIPassByRef("some_value")
var2 = LMIPassByRef("some_other_value")

def indication_callback(ind, var1, var2):
    # Do something with ind, var1 and var2
    print ind.exported_objects()
    print var1.value
    print var2.value

c = connect("hostname", "username", "password")

listener = LMIIIndicationListener("0.0.0.0", 65500)
unique_name = listener.add_handler(
    "demo-XXXXXXX",      # Creates a unique name for me
    indication_callback, # Callback to be called
    var1,                # Variable passed by ref
    var2                 # Variable passed by ref
)

listener.start()

print c.subscribe_indication(
    Name=unique_name,
    Query="SELECT * FROM LMI_AccountInstanceCreationIndication WHERE
SOURCEINSTANCE ISA LMI_Account",
    Destination="192.168.122.1:65500"
)

try:
    while True:
        sleep(60)
except KeyboardInterrupt:
    sys.exit(0)
```

17.4.10. Example Usage

This section provides a number of examples for various CIM providers distributed with the OpenLMI packages. All examples in this section use the following two variable definitions:

```
c = connect("host_name", "user_name", "password")
ns = c.root.cimv2
```

Replace *host_name* with the host name of the managed system, *user_name* with the name of user that is allowed to connect to OpenPegasus CIMOM running on that system, and *password* with the user's password.

Using the OpenLMI Service Provider

The *openlmi-service* package installs a CIM provider for managing system services. The examples below illustrate how to use this CIM provider to list available system services and how to start, stop, enable, and disable them.

Example 17.41. Listing Available Services

To list all available services on the managed machine along with information regarding whether the service has been started (**TRUE**) or stopped (**FALSE**) and the status string, use the following code snippet:

```
for service in ns.LMI_Service.instances():
    print "%s:\t%s" % (service.Name, service.Status)
```

To list only the services that are enabled by default, use this code snippet:

```
cls = ns.LMI_Service
for service in cls.instances():
    if service.EnabledDefault == cls.EnabledDefaultValues.Enabled:
        print service.Name
```

Note that the value of the **EnabledDefault** property is equal to **2** for enabled services and **3** for disabled services.

To display information about the **cups** service, use the following:

```
cups = ns.LMI_Service.first_instance({"Name": "cups.service"})
cups.doc()
```

Example 17.42. Starting and Stopping Services

To start and stop the **cups** service and to see its current status, use the following code snippet:

```
cups = ns.LMI_Service.first_instance({"Name": "cups.service"})
cups.StartService()
print cups.Status
cups.StopService()
print cups.Status
```

Example 17.43. Enabling and Disabling Services

To enable and disable the **cups** service and to display its **EnabledDefault** property, use the following code snippet:

```
cups = ns.LMI_Service.first_instance({"Name": "cups.service"})
cups.TurnServiceOff()
print cups.EnabledDefault
cups.TurnServiceOn()
print cups.EnabledDefault
```

Using the OpenLMI Networking Provider

The *openlmi-networking* package installs a CIM provider for networking. The examples below illustrate how to use this CIM provider to list IP addresses associated with a certain port number, create a new connection, configure a static IP address, and activate a connection.

Example 17.44. Listing IP Addresses Associated with a Given Port Number

To list all IP addresses associated with the eth0 network interface, use the following code snippet:

```
device = ns.LMI_IPNetworkConnection.first_instance({'ElementName': 'eth0'})
for endpoint in device.associators(AssocClass="LMI_NetworkSAPSAPDependency",
ResultClass="LMI_IPProtocolEndpoint"):
    if endpoint.ProtocolIFTType ==
        ns.LMI_IPProtocolEndpoint.ProtocolIFTTypeValues.IPV4:
            print "IPv4: %s/%s" % (endpoint.IPV4Address, endpoint.SubnetMask)
    elif endpoint.ProtocolIFTType ==
        ns.LMI_IPProtocolEndpoint.ProtocolIFTTypeValues.IPV6:
            print "IPv6: %s/%d" % (endpoint.IPV6Address,
endpoint.IPV6SubnetPrefixLength)
```

This code snippet uses the **LMI_IPProtocolEndpoint** class associated with a given **LMI_IPNetworkConnection** class.

To display the default gateway, use this code snippet:

```
for rsap in
device.associators(AssocClass="LMI_NetworkRemoteAccessAvailableToElement",
ResultClass="LMI_NetworkRemoteServiceAccessPoint"):
    if rsap.AccessContext ==
        ns.LMI_NetworkRemoteServiceAccessPoint.AccessContextValues.DefaultGateway:
            print "Default Gateway: %s" % rsap.AccessInfo
```

The default gateway is represented by an **LMI_NetworkRemoteServiceAccessPoint** instance with the **AccessContext** property equal to **DefaultGateway**.

To get a list of DNS servers, the object model needs to be traversed as follows:

1. Get the **LMI_IPProtocolEndpoint** instances associated with a given **LMI_IPNetworkConnection** using **LMI_NetworkSAPSAPDependency**.
2. Use the same association for the **LMI_DNSProtocolEndpoint** instances.

The **LMI_NetworkRemoteServiceAccessPoint** instances with the **AccessContext** property equal to the DNS Server associated through **LMI_NetworkRemoteAccessAvailableToElement** have the DNS server address in the **AccessInfo** property.

There can be more possible paths to get to the **RemoteServiceAccessPath** and entries can be duplicated. The following code snippet uses the **set()** function to remove duplicate entries from the list of DNS servers:

```
dnsservers = set()
for ipendpoint in device.associators(AssocClass="LMI_NetworkSAPSAPDependency",
ResultClass="LMI_IPProtocolEndpoint"):
    for dnsedpoint in
ipendpoint.associators(AssocClass="LMI_NetworkSAPSAPDependency",
ResultClass="LMI_DNSProtocolEndpoint"):
        for rsap in
dnsedpoint.associators(AssocClass="LMI_NetworkRemoteAccessAvailableToElement",
ResultClass="LMI_NetworkRemoteServiceAccessPoint"):
            if rsap.AccessContext == ns.LMI_NetworkRemoteServiceAccessPoint.AccessContextValues.DNSServer:
                dnsservers.add(rsap.AccessInfo)
print "DNS:", ", ".join(dnsservers)
```

Example 17.45. Creating a New Connection and Configuring a Static IP Address

To create a new setting with a static IPv4 and stateless IPv6 configuration for network interface eth0, use the following code snippet:

```
capability = ns.LMIPNetworkConnectionCapabilities.first_instance({ 'ElementName' :
'eth0' })
result = capability.LMI_CreateIPSetting(Caption='eth0 Static',
IPv4Type=capability.LMI_CreateIPSetting.IPv4TypeValues.Static,
IPv6Type=capability.LMI_CreateIPSetting.IPv6TypeValues.Stateless)
setting = result.rparams["SettingData"].to_instance()
for settingData in
setting.associators(AssocClass="LMI_OrderedIPAssignmentComponent"):
    if setting.ProtocolIFTType ==
ns.LMIPAssignmentSettingData.ProtocolIFTTypeValues.IPV4:
        # Set static IPv4 address
        settingData.IPAddresses = ["192.168.1.100"]
        settingData.SubnetMasks = ["255.255.0.0"]
        settingData.GatewayAddresses = ["192.168.1.1"]
        settingData.push()
```

This code snippet creates a new setting by calling the **LMI_CreateIPSetting()** method on the instance of **LMI_IPNetworkConnectionCapabilities**, which is associated with **LMI_IPNetworkConnection** through **LMI_IPNetworkConnectionElementCapabilities**. It also uses the **push()** method to modify the setting.

Example 17.46. Activating a Connection

To apply a setting to the network interface, call the `ApplySettingToIPNetworkConnection()` method of the `LMI_IPConfigurationService` class. This method is asynchronous and returns a job. The following code snippets illustrates how to call this method synchronously:

```
setting = ns.LMI_IPAssignmentSettingData.first_instance({ "Caption": "eth0 Static" })
port = ns.LMI_IPNetworkConnection.first_instance({ 'ElementName': 'ens8' })
service = ns.LMI_IPConfigurationService.first_instance()
service.SyncApplySettingToIPNetworkConnection(SettingData=setting,
IPNetworkConnection=port, Mode=32768)
```

The **Mode** parameter affects how the setting is applied. The most commonly used values of this parameter are as follows:

- » **1** — apply the setting now and make it auto-activated.
- » **2** — make the setting auto-activated and do not apply it now.
- » **4** — disconnect and disable auto-activation.
- » **5** — do not change the setting state, only disable auto-activation.
- » **32768** — apply the setting.
- » **32769** — disconnect.

Using the OpenLMI Storage Provider

The `openlmi-storage` package installs a CIM provider for storage management. The examples below illustrate how to use this CIM provider to create a volume group, create a logical volume, build a file system, mount a file system, and list block devices known to the system.

In addition to the `c` and `ns` variables, these examples use the following variable definitions:

```
MEGABYTE = 1024*1024
storage_service = ns.LMI_StorageConfigurationService.first_instance()
filesystem_service = ns.LMI_FileSystemConfigurationService.first_instance()
```

Example 17.47. Creating a Volume Group

To create a new volume group located in `/dev/myGroup/` that has three members and the default extent size of 4 MB, use the following code snippet:

```
# Find the devices to add to the volume group
# (filtering the CIM_StorageExtent.instances())
# call would be faster, but this is easier to read):
sda1 = ns.CIM_StorageExtent.first_instance({"Name": "/dev/sda1"})
sdb1 = ns.CIM_StorageExtent.first_instance({"Name": "/dev/sdb1"})
sdc1 = ns.CIM_StorageExtent.first_instance({"Name": "/dev/sdc1"})

# Create a new volume group:
(ret, outparams, err) = storage_service.SyncCreateOrModifyVG(
    ElementName="myGroup",
    InExtents=[sda1, sdb1, sdc1])
vg = outparams['Pool'].to_instance()
print "VG", vg.PoolID, \
    "with extent size", vg.ExtentSize, \
    "and", vg.RemainingExtents, "free extents created."
```

Example 17.48. Creating a Logical Volume

To create two logical volumes with the size of 100 MB, use this code snippet:

```
# Find the volume group:
vg = ns.LMI_VGStoragePool.first_instance({"Name": "/dev/mapper/myGroup"})

# Create the first logical volume:
(ret, outparams, err) = storage_service.SyncCreateOrModifyLV(
    ElementName="Vol1",
    InPool=vg,
    Size=100 * MEGABYTE)
lv = outparams['TheElement'].to_instance()
print "LV", lv.DeviceID, \
    "with", lv.BlockSize * lv.NumberOfBlocks, \
    "bytes created."

# Create the second logical volume:
(ret, outparams, err) = storage_service.SyncCreateOrModifyLV(
    ElementName="Vol2",
    InPool=vg,
    Size=100 * MEGABYTE)
lv = outparams['TheElement'].to_instance()
print "LV", lv.DeviceID, \
    "with", lv.BlockSize * lv.NumberOfBlocks, \
    "bytes created."
```

Example 17.49. Creating a File System

To create an **ext3** file system on logical volume **lv** from [Example 17.48, “Creating a Logical Volume”](#), use the following code snippet:

```
(ret, outparams, err) = filesystem_service.SyncLMI_CreateFileSystem(
    FileSystemType=filesystem_service.LMI_CreateFileSystem.FileSystemTypeValues.EXT3,
    InExtents=[lv])
```

Example 17.50. Mounting a File System

To mount the file system created in [Example 17.49, “Creating a File System”](#), use the following code snippet:

```
# Find the file system on the logical volume:
fs = lv.first_associator(ResultClass="LMI_LocalFileSystem")

mount_service = ns.LMI_MountConfigurationService.first_instance()
(rc, out, err) = mount_service.SyncCreateMount(
    FileSystemType='ext3',
    Mode=32768, # just mount
    FileSystem=fs,
    MountPoint='/mnt/test',
    FileSystemSpec=lv.Name)
```

Example 17.51. Listing Block Devices

To list all block devices known to the system, use the following code snippet:

```
devices = ns.CIM_StorageExtent.instances()
for device in devices:
    if lmi_isinstance(device, ns.CIM_Memory):
        # Memory and CPU caches are StorageExtents too, do not print them
        continue
    print device.classname,
    print device.DeviceID,
    print device.Name,
    print device.BlockSize*device.NumberOfBlocks
```

Using the OpenLMI Hardware Provider

The *openlmi-hardware* package installs a CIM provider for monitoring hardware. The examples below illustrate how to use this CIM provider to retrieve information about CPU, memory modules, PCI devices, and the manufacturer and model of the machine.

Example 17.52. Viewing CPU Information

To display basic CPU information such as the CPU name, the number of processor cores, and the number of hardware threads, use the following code snippet:

```
cpu = ns.LMI_Processor.first_instance()
cpu_cap = cpu.associators(ResultClass="LMI_ProcessorCapabilities")[0]
print cpu.Name
print cpu_cap.NumberOfProcessorCores
print cpu_cap.NumberOfHardwareThreads
```

Example 17.53. Viewing Memory Information

To display basic information about memory modules such as their individual sizes, use the following code snippet:

```
mem = ns.LMI_Memory.first_instance()
for i in mem.associators(ResultClass="LMI_PhysicalMemory"):
    print i.Name
```

Example 17.54. Viewing Chassis Information

To display basic information about the machine such as its manufacturer or its model, use the following code snippet:

```
chassis = ns.LMI_Chassis.first_instance()
print chassis.Manufacturer
print chassis.Model
```

Example 17.55. Listing PCI Devices

To list all PCI devices known to the system, use the following code snippet:

```
for pci in ns.LMI_PCIDevice.instances():
    print pci.Name
```

17.5. Using OpenLMI Scripts

The LMIShell interpreter is built on top of Python modules that can be used to develop custom management tools. The OpenLMI Scripts project provides a number of Python libraries for interfacing with OpenLMI providers. In addition, it is distributed with **lmi**, an extensible utility that can be used to interact with these libraries from the command line.

To install OpenLMI Scripts on your system, type the following at a shell prompt:

```
easy_install --user openlmi-scripts
```

This command installs the Python modules and the **lmi** utility in the `~/.local/` directory. To extend the functionality of the **lmi** utility, install additional OpenLMI modules by using the following command:

```
easy_install --user package_name
```

For a complete list of available modules, see the [Python website](#). For more information about OpenLMI Scripts, see the [official OpenLMI Scripts documentation](#).

17.6. Additional Resources

For more information about OpenLMI and system management in general, see the resources listed below.

Installed Documentation

- ▶ **lmishell(1)** — The manual page for the **lmishell** client and interpreter provides detailed information about its execution and usage.

Online Documentation

- ▶ [Red Hat Enterprise Linux 7 Networking Guide](#) — The *Networking Guide* for Red Hat Enterprise Linux 7 documents relevant information regarding the configuration and administration of network interfaces and network services on the system.
- ▶ [Red Hat Enterprise Linux 7 Storage Administration Guide](#) — The *Storage Administration Guide* for Red Hat Enterprise Linux 7 provides instructions on how to manage storage devices and file systems on the system.
- ▶ [Red Hat Enterprise Linux 7 Power Management Guide](#) — The *Power Management Guide* for Red Hat Enterprise Linux 7 explains how to manage power consumption of the system effectively. It discusses different techniques that lower power consumption for both servers and laptops, and explains how each technique affects the overall performance of the system.
- ▶ [Red Hat Enterprise Linux 7 Linux Domain Identity, Authentication, and Policy Guide](#) — The *Linux Domain Identity, Authentication, and Policy Guide* for Red Hat Enterprise Linux 7 covers all aspects of installing, configuring, and managing IPA domains, including both servers and clients. The guide is intended for IT and systems administrators.
- ▶ [FreeIPA Documentation](#) — The *FreeIPA Documentation* serves as the primary user documentation for using the FreeIPA Identity Management project.
- ▶ [OpenSSL Home Page](#) — The *OpenSSL* home page provides an overview of the OpenSSL project.
- ▶ [Mozilla NSS Documentation](#) — The *Mozilla NSS Documentation* serves as the primary user documentation for using the Mozilla NSS project.

See Also

- ▶ [Chapter 3, Managing Users and Groups](#) documents how to manage system users and groups in the graphical user interface and on the command line.
- ▶ [Chapter 5, Yum](#) describes how to use the **Yum** package manager to search, install, update, and uninstall packages on the command line.
- ▶ [Chapter 6, Managing Services with systemd](#) provides an introduction to **systemd** and documents how to use the **systemctl** command to manage system services, configure systemd targets, and execute power management commands.
- ▶ [Chapter 7, OpenSSH](#) describes how to configure an SSH server and how to use the **ssh**, **scp**, and **sftp** client utilities to access it.

Chapter 18. Viewing and Managing Log Files

Log files are files that contain messages about the system, including the kernel, services, and applications running on it. There are several types of log files for storing various information. For example, there is a default system log file, a log file just for security messages, and a log file for cron tasks. Log files can be very useful in many situations, for instance to troubleshoot a problem with the system, when trying to load a kernel driver, or when looking for unauthorized login attempts to the system.

Some log files are controlled by a daemon called **rsyslogd**. The **rsyslogd** daemon is an enhanced replacement for previous **syslogd**, and provides extended filtering, various configuration options, input and output modules, support for transportation via the **TCP** or **UDP** protocols. A list of log files maintained by **rsyslogd** can be found in the **/etc/rsyslog.conf** configuration file. Most log files are located in the **/var/log/** directory.

Log files can also be managed by the **journald** daemon – a component of **systemd**. The **journald** daemon captures Syslog messages, kernel log messages, initial RAM disk and early boot messages as well as messages written to standard output and standard error output of all services, indexes them and makes this available to the user. The native journal file format, which is a structured and indexed binary file, improves searching and provides faster operation, and it also stores meta data information like timestamps or user IDs. Log files produced by **journald** are by default not persistent, log files are stored only in memory or a small ring-buffer in the **/run/log/journal/** directory. The amount of logged data depends on free memory, when you reach the capacity limit, the oldest entries are deleted. However, this setting can be altered – see [Section 18.8.5, “Enabling Persistent Storage”](#). For more information on Journal see [Section 18.8, “Using the Journal”](#).

By default, these two logging tools coexist on your system. The additional structured data that is added to messages by the **journald** daemon that is also the primary tool for troubleshooting. Data acquired by **journald** are forwarded into the **/run/systemd/journal/syslog** socket that may be used by **rsyslogd** to process the data further. However, **rsyslog** does the actual integration by default via the **imjournal** input module, thus avoiding the aforementioned socket. You can also transfer data in the opposite direction, from **rsyslogd** to **journald** with use of **omjournal** module. See [Section 18.5, “Interaction of Rsyslog and Journal”](#) for further information. The integration allows to maintain text-based logs in a consistent format to assure compatibility with possible applications or configurations dependent on **rsyslogd**. Also, you can maintain rsyslog messages in a structured format (see [Section 18.6, “Structured Logging with Rsyslog”](#)).

18.1. Locating Log Files

Most log files are located in the **/var/log/** directory. Some applications such as **httpd** and **samba** have a directory within **/var/log/** for their log files.

You may notice multiple files in the **/var/log/** directory with numbers after them (for example, **cron-20100906**). These numbers represent a time stamp that has been added to a rotated log file. Log files are rotated so their file sizes do not become too large. The **logrotate** package contains a cron task that automatically rotates log files according to the **/etc/logrotate.conf** configuration file and the configuration files in the **/etc/logrotate.d/** directory.

18.2. Basic Configuration of Rsyslog

The main configuration file for **rsyslog** is **/etc/rsyslog.conf**. Here, you can specify *global directives*, *modules*, and *rules* that consist of *filter* and *action* parts. Also, you can add comments in form of text following a hash sign (#).

18.2.1. Filters

A rule is specified by a *filter* part, which selects a subset of syslog messages, and an *action* part, which specifies what to do with the selected messages. To define a rule in your `/etc/rsyslog.conf` configuration file, define both, a filter and an action, on one line and separate them with one or more spaces or tabs.

rsyslog offers various ways to filter syslog messages according to selected properties. The available filtering methods can be divided into *Facility/Priority-based*, *Property-based*, and *Expression-based* filters.

Facility/Priority-based filters

The most used and well-known way to filter syslog messages is to use the facility/priority-based filters which filter syslog messages based on two conditions: *facility* and *priority* separated by a comma. To create a selector, use the following syntax:

`FACILITY.PRIORITY`

where:

- ▶ *FACILITY* specifies the subsystem that produces a specific syslog message. For example, the **mail** subsystem handles all mail-related syslog messages. *FACILITY* can be represented by one of the following keywords: **auth**, **authpriv**, **cron**, **daemon**, **kern**, **lpr**, **mail**, **news**, **syslog**, **user**, **ftp**, **uucp**, and **local0** through **local7**.
- ▶ *PRIORITY* specifies a priority of a syslog message. *PRIORITY* can be represented by one of the following keywords (or by a number): **debug** (7), **info** (6), **notice** (5), **warning** (4), **err** (3), **crit** (2), **alert** (1), and **emerg** (0).

The aforementioned syntax selects syslog messages with the defined or *higher* priority. By preceding any priority keyword with an equal sign (=), you specify that only syslog messages with the specified priority will be selected. All other priorities will be ignored. Conversely, preceding a priority keyword with an exclamation mark (!) selects all syslog messages except those with the defined priority.

In addition to the keywords specified above, you may also use an asterisk (*) to define all facilities or priorities (depending on where you place the asterisk, before or after the comma). Specifying the priority keyword **none** serves for facilities with no given priorities. Both facility and priority conditions are case-insensitive.

To define multiple facilities and priorities, separate them with a comma (,). To define multiple selectors on one line, separate them with a semi-colon (;). Note that each selector in the selector field is capable of overwriting the preceding ones, which can exclude some priorities from the pattern.

Example 18.1. Facility/Priority-based Filters

The following are a few examples of simple facility/priority-based filters that can be specified in `/etc/rsyslog.conf`. To select all kernel syslog messages with any priority, add the following text into the configuration file:

```
kern.*
```

To select all mail syslog messages with priority **`crit`** and higher, use this form:

```
mail.crit
```

To select all cron syslog messages except those with the **`info`** or **`debug`** priority, set the configuration in the following form:

```
cron.!info, !debug
```

Property-based filters

Property-based filters let you filter syslog messages by any property, such as **`timegenerated`** or **`syslogtag`**. For more information on properties, see [Section 18.2.3, “Properties”](#). You can compare each of the specified properties to a particular value using one of the compare-operations listed in [Table 18.1, “Property-based compare-operations”](#). Both property names and compare operations are case-sensitive.

Property-based filter must start with a colon (:). To define the filter, use the following syntax:

```
:PROPERTY, [!]COMPARE_OPERATION, "STRING"
```

where:

- ▶ The **`PROPERTY`** attribute specifies the desired property.
- ▶ The optional exclamation point (!) negates the output of the compare-operation. Other Boolean operators are currently not supported in property-based filters.
- ▶ The **`COMPARE_OPERATION`** attribute specifies one of the compare-operations listed in [Table 18.1, “Property-based compare-operations”](#).
- ▶ The **`STRING`** attribute specifies the value that the text provided by the property is compared to. This value must be enclosed in quotation marks. To escape certain character inside the string (for example a quotation mark (")), use the backslash character (\).

Table 18.1. Property-based compare-operations

Compare-operation	Description
<code>contains</code>	Checks whether the provided string matches any part of the text provided by the property. To perform case-insensitive comparisons, use <code>contains_i</code> .
<code>isequal</code>	Compares the provided string against all of the text provided by the property. These two values must be exactly equal to match.

Compare-operation	Description
startswith	Checks whether the provided string is found exactly at the beginning of the text provided by the property. To perform case-insensitive comparisons, use startswith_i .
regex	Compares the provided POSIX BRE (Basic Regular Expression) regular expression against the text provided by the property.
ereregex	Compares the provided POSIX ERE (Extended Regular Expression) regular expression against the text provided by the property.
isempty	Checks if the property is empty. The value is discarded. This is especially useful when working with normalized data, where some fields may be populated based on normalization result.

Example 18.2. Property-based Filters

The following are a few examples of property-based filters that can be specified in **/etc/rsyslog.conf**. To select syslog messages which contain the string **error** in their message text, use:

```
:msg, contains, "error"
```

The following filter selects syslog messages received from the host name **host1**:

```
:hostname, isequal, "host1"
```

To select syslog messages which do not contain any mention of the words **fatal** and **error** with any or no text between them (for example, **fatal lib error**), type:

```
:msg, !regex, "fatal .* error"
```

Expression-based filters

Expression-based filters select syslog messages according to defined arithmetic, Boolean or string operations. Expression-based filters use **rsyslog**'s own scripting language called *RainerScript* to build complex filters. See [Section 18.10, “Online Documentation”](#) for the syntax definition of this script along with examples of various expression-based filters. Also RainerScript is a basis for **rsyslog**'s new configuration format, see [Section 18.2.6, “Using the New Configuration Format”](#).

The basic syntax of expression-based filter looks as follows:

```
if EXPRESSION then ACTION else ACTION
```

where:

- ▶ The **EXPRESSION** attribute represents an expression to be evaluated, for example: **\$msg startswith 'DEVNAME'** or **\$syslogfacility-text == 'local0'**. You can specify more than one expression in single filter by using and/or operators.
- ▶ The **ACTION** attribute represents an action to be performed if the expression returns the value **true**. This can be a single action, or an arbitrary complex script enclosed in curly braces.

- ▶ Expression-based filters are indicated by the keyword `if` at the start of a new line. The `then` keyword separates the *EXPRESSION* from the *ACTION*. Optionally, you can employ the `else` keyword to specify what action is to be performed in case the condition is not met.

With expression-based filters, you can nest the conditions by using a script enclosed in curly braces as in [Example 18.3, “Expression-based Filters”](#). The script allows you to use *facility/priority-based* filters inside the expression. On the other hand, *property-based* filters are not recommended here. RainerScript supports regular expressions with specialized functions `re_match()` and `re_extract()`

Example 18.3. Expression-based Filters

The following expression contains two nested conditions. The log files created by a program called `prog1` are split into two files based on the presence of the "test" string in the message.

```
if $programname == 'prog1' then {
    action(type="omfile" file="/var/log/prog1.log")
    if $msg contains 'test' then
        action(type="omfile" file="/var/log/prog1test.log")
    else
        action(type="omfile" file="/var/log/prog1notest.log")
}
```

18.2.2. Actions

Actions specify what is to be done with the messages filtered out by an already-defined selector. The following are some of the actions you can define in your rule:

Saving syslog messages to log files

The majority of actions specify to which log file a syslog message is saved. This is done by specifying a file path after your already-defined selector:

FILTER PATH

where *FILTER* stands for user-specified selector and *PATH* is a path of a target file.

For instance, the following rule is comprised of a selector that selects all **cron** syslog messages and an action that saves them into the `/var/log/cron.log` log file:

`cron.* /var/log/cron.log`

By default, the log file is synchronized every time a syslog message is generated. Use a dash mark (-) as a prefix of the file path you specified to omit syncing:

FILTER -PATH

Note that you might lose information if the system terminates right after a write attempt. However, this setting can save some performance, especially if you run programs that produce very verbose log messages.

Your specified file path can be either *static* or *dynamic*. Static files are represented by a fixed file path as was shown in the example above. Dynamic file paths can differ according to the received message. Dynamic file paths are represented by a template and a question mark (?) prefix:

`FILTER ?DynamicFile`

where *DynamicFile* is a name of a predefined template that modifies output paths. You can use the dash prefix (-) to disable syncing, also you can use multiple templates separated by colon (:). For more information on templates, see [Section 18.2.3, “Generating Dynamic File Names”](#).

If the file you specified is an existing **terminal** or **/dev/console** device, syslog messages are sent to standard output (using special **terminal**-handling) or your console (using special **/dev/console**-handling) when using the X Window System, respectively.

Sending syslog messages over the network

rsyslog allows you to send and receive syslog messages over the network. This feature allows you to administer syslog messages of multiple hosts on one machine. To forward syslog messages to a remote machine, use the following syntax:

`@[(zNUMBER)]HOST:[PORT]`

where:

- ▶ The at sign (@) indicates that the syslog messages are forwarded to a host using the **UDP** protocol. To use the **TCP** protocol, use two at signs with no space between them (@@).
- ▶ The optional **zNUMBER** setting enables **zlib** compression for syslog messages. The **NUMBER** attribute specifies the level of compression (from 1 – lowest to 9 – maximum). Compression gain is automatically checked by **rsyslogd**, messages are compressed only if there is any compression gain and messages below 60 bytes are never compressed.
- ▶ The **HOST** attribute specifies the host which receives the selected syslog messages.
- ▶ The **PORT** attribute specifies the host machine's port.

When specifying an **IPv6** address as the host, enclose the address in square brackets ([,]).

Example 18.4. Sending syslog Messages over the Network

The following are some examples of actions that forward syslog messages over the network (note that all actions are preceded with a selector that selects all messages with any priority). To forward messages to **192.168.0.1** via the **UDP** protocol, type:

`*.* @192.168.0.1`

To forward messages to "example.com" using port 18 and the **TCP** protocol, use:

`*.* @@example.com:18`

The following compresses messages with **zlib** (level 9 compression) and forwards them to **2001:db8::1** using the **UDP** protocol

`*.* @z9[2001:db8::1]`

Output channels

Output channels are primarily used to specify the maximum size a log file can grow to. This is very useful for log file rotation (for more information see [Section 18.2.5, “Log Rotation”](#)). An output channel is basically a collection of information about the output action. Output channels are defined by the **\$outchannel** directive. To define an output channel in **/etc/rsyslog.conf**, use the following syntax:

```
$outchannel NAME, FILE_NAME, MAX_SIZE, ACTION
```

where:

- ▶ The *NAME* attribute specifies the name of the output channel.
- ▶ The *FILE_NAME* attribute specifies the name of the output file. Output channels can write only into files, not pipes, terminal or other kind of output.
- ▶ The *MAX_SIZE* attribute represents the maximum size the specified file (in *FILE_NAME*) can grow to. This value is specified in bytes.
- ▶ The *ACTION* attribute specifies the action that is taken when the maximum size, defined in *MAX_SIZE*, is hit.

To use the defined output channel can as an action inside a rule, type:

```
FILTER :omfile:$NAME
```

Example 18.5. Output channel log rotation

The following output shows a simple log rotation through the use of an output channel. First, the output channel is defined via the **\$outchannel** directive:

```
$outchannel log_rotation, /var/log/test_log.log, 104857600,  
/home/joe/log_rotation_script
```

and then it is used in a rule that selects every syslog message with any priority and executes the previously-defined output channel on the acquired syslog messages:

```
*.* :omfile:$log_rotation
```

Once the limit (in the example **100 MB**) is hit, the **/home/joe/log_rotation_script** is executed. This script can contain anything from moving the file into a different folder, editing specific content out of it, or simply removing it.

Sending syslog messages to specific users

rsyslog can send syslog messages to specific users by specifying a user name of the user you wish to send the messages to (as in [Example 18.7, “Specifying Multiple Actions”](#)). To specify more than one user, separate each user name with a comma (,). To send messages to every user that is currently logged on, use an asterisk (*).

Executing a program

rsyslog lets you execute a program for selected syslog messages and uses the **system()** call to execute the program in shell. To specify a program to be executed, prefix it with a caret character (^). Consequently, specify a template that formats the received message and passes it to the specified executable as a one line parameter (for more information on templates, see [Section 18.2.3, “Templates”](#)).

```
FILTER ^EXECUTABLE; TEMPLATE
```

Here an output of the *FILTER* condition is processed by a program represented by *EXECUTABLE*. This program can be any valid executable. Replace *TEMPLATE* with the name of the formating template.

Example 18.6. Executing a Program

In the following example, any syslog message with any priority is selected, formatted with the *template* template and passed as a parameter to the **test-program** program, which is then executed with the provided parameter:

```
*.* ^test-program;template
```



Be careful when using the shell execute action

When accepting messages from any host, and using the shell execute action, you may be vulnerable to command injection. An attacker may try to inject and execute his commands in the program you specified to be executed in your action. To avoid any possible security threats, thoroughly consider the use of the shell execute action.

Storing syslog messages in a database

Selected syslog messages can be directly written into a database table using the *database writer* action. The database writer uses the following syntax:

```
:PLUGIN:DB_HOST, DB_NAME, DB_USER, DB_PASSWORD;[ TEMPLATE]
```

where:

- ▶ The *PLUGIN* calls the specified plug-in that handles the database writing (for example, the **omm.mysql** plug-in).
- ▶ The *DB_HOST* attribute specifies the database host name.
- ▶ The *DB_NAME* attribute specifies the name of the database.
- ▶ The *DB_USER* attribute specifies the database user.
- ▶ The *DB_PASSWORD* attribute specifies the password used with the aforementioned database user.
- ▶ The *TEMPLATE* attribute specifies an optional use of a template that modifies the syslog message. For more information on templates, see [Section 18.2.3, “Templates”](#).



Using MySQL and PostgreSQL

Currently, **rsyslog** provides support for **MySQL** and **PostgreSQL** databases only. In order to use the **MySQL** and **PostgreSQL** database writer functionality, install the *rsyslog-mysql* and *rsyslog-pgsql* packages, respectively. Also, make sure you load the appropriate modules in your **/etc/rsyslog.conf** configuration file:

```
$ModLoad ommysql      # Output module for MySQL support
$ModLoad ompgsql       # Output module for PostgreSQL support
```

For more information on **rsyslog** modules, see [Section 18.4, “Using Rsyslog Modules”](#).

Alternatively, you may use a generic database interface provided by the **omlibdb** module (supports: Firebird/Interbase, MS SQL, Sybase, SQLite, Ingres, Oracle, mSQL).

Discarding syslog messages

To discard your selected messages, use the tilde character (~).

FILTER ~

The discard action is mostly used to filter out messages before carrying on any further processing. It can be effective if you want to omit some repeating messages that would otherwise fill the log files. The results of discard action depend on where in the configuration file it is specified, for the best results place these actions on top of the actions list. Please note that once a message has been discarded there is no way to retrieve it in later configuration file lines.

For instance, the following rule discards any cron syslog messages:

cron.* ~

Specifying Multiple Actions

For each selector, you are allowed to specify multiple actions. To specify multiple actions for one selector, write each action on a separate line and precede it with an ampersand character (&):

FILTER ACTION
& *ACTION*
& *ACTION*

Specifying multiple actions improves the overall performance of the desired outcome since the specified selector has to be evaluated only once.

Example 18.7. Specifying Multiple Actions

In the following example, all kernel syslog messages with the critical priority (***crit***) are sent to user **user1**, processed by the template ***temp*** and passed on to the ***test-program*** executable, and forwarded to **192.168.0.1** via the **UDP** protocol.

```
kern.=crit user1
& ^test-program;temp
& @192.168.0.1
```

Any action can be followed by a template that formats the message. To specify a template, suffix an action with a semicolon (;) and specify the name of the template. For more information on templates, see [Section 18.2.3, “Templates”](#).



Using templates

A template must be defined before it is used in an action, otherwise it is ignored. In other words, template definitions should always precede rule definitions in **/etc/rsyslog.conf**.

18.2.3. Templates

Any output that is generated by **rsyslog** can be modified and formatted according to your needs with the use of **templates**. To create a template use the following syntax in **/etc/rsyslog.conf**:

```
$template TEMPLATE_NAME,"text %PROPERTY% more text", [OPTION]
```

where:

- ▶ **\$template** is the template directive that indicates that the text following it, defines a template.
- ▶ **TEMPLATE_NAME** is the name of the template. Use this name to refer to the template.
- ▶ Anything between the two quotation marks ("... ") is the actual template text. Within this text, special characters, such as **\n** for new line or **\r** for carriage return, can be used. Other characters, such as **%** or **"**, have to be escaped if you want to use those characters literally.
- ▶ The text specified within two percent signs (%) specifies a *property* that allows you to access specific contents of a syslog message. For more information on properties, see [Section 18.2.3, “Properties”](#).
- ▶ The **OPTION** attribute specifies any options that modify the template functionality. The currently supported template options are **sql** and **stdsql**, which are used for formatting the text as an SQL query.



The sql and stdsql options

Note that the database writer (for more information, see section *Storing syslog messages in a database* in [Section 18.2.2, “Actions”](#)) checks whether the **sql** or **stdsql** options are specified in the template. If they are not, the database writer does not perform any action. This is to prevent any possible security threats, such as SQL injection.

Generating Dynamic File Names

Templates can be used to generate dynamic file names. By specifying a property as a part of the file path, a new file will be created for each unique property, which is a convenient way to classify syslog messages.

For example, use the ***timegenerated*** property, which extracts a time stamp from the message, to generate a unique file name for each syslog message:

```
$template DynamicFile, "/var/log/test_logs/%timegenerated%-test.log"
```

Keep in mind that the **\$template** directive only specifies the template. You must use it inside a rule for it to take effect. In **/etc/rsyslog.conf**, use the question mark (?) in action definition to mark the dynamic filename template:

```
*.* ?DynamicFile
```

Properties

Properties defined inside a template (within two percent signs (%)) allow to access various contents of a syslog message through the use of a *property replacer*. To define a property inside a template (between the two quotation marks ("...")), use the following syntax:

```
%PROPERTY_NAME[ :FROM_CHAR:TO_CHAR:OPTION]%
```

where:

- ▶ The **PROPERTY_NAME** attribute specifies the name of a property. A list of all available properties and their detailed description can be found in the **rsyslog.conf(5)** manual page under the section *Available Properties*.
- ▶ **FROM_CHAR** and **TO_CHAR** attributes denote a range of characters that the specified property will act upon. Alternatively, regular expressions can be used to specify a range of characters. To do so, set the letter **R** as the **FROM_CHAR** attribute and specify your desired regular expression as the **TO_CHAR** attribute.
- ▶ The **OPTION** attribute specifies any property options, such as the **lowercase** option to convert the input to lowercase. A list of all available property options and their detailed description can be found in the **rsyslog.conf(5)** manual page under the section *Property Options*.

The following are some examples of simple properties:

- ▶ The following property obtains the whole message text of a syslog message:

```
%msg%
```

- ▶ The following property obtains the first two characters of the message text of a syslog message:

```
%msg:1:2%
```

- ▶ The following property obtains the whole message text of a syslog message and drops its last line feed character:

```
%msg:::drop-last-lf%
```

- The following property obtains the first 10 characters of the time stamp that is generated when the syslog message is received and formats it according to the [RFC 3999](#) date standard.

```
%timegenerated:1:10:date-rfc3339%
```

Template Examples

This section presents few examples of **rsyslog** templates.

[Example 18.8, “A verbose syslog message template”](#) shows a template that formats a syslog message so that it outputs the message's severity, facility, the time stamp of when the message was received, the host name, the message tag, the message text, and ends with a new line.

Example 18.8. A verbose syslog message template

```
$template verbose, "%syslogseverity%, %syslogfacility%, %timegenerated%,  
%HOSTNAME%, %syslogtag%, %msg%\n"
```

[Example 18.9, “A wall message template”](#) shows a template that resembles a traditional wall message (a message that is send to every user that is logged in and has their **msg(1)** permission set to **yes**). This template outputs the message text, along with a host name, message tag and a time stamp, on a new line (using **\r** and **\n**) and rings the bell (using **\7**).

Example 18.9. A wall message template

```
$template wallmsg, "\r\n\7Message from syslogd@%HOSTNAME% at %timegenerated%  
...\r\n %syslogtag% %msg%\n\r"
```

[Example 18.10, “A database formatted message template”](#) shows a template that formats a syslog message so that it can be used as a database query. Notice the use of the **sql** option at the end of the template specified as the template option. It tells the database writer to format the message as an MySQL **SQL** query.

Example 18.10. A database formatted message template

```
$template dbFormat,"insert into SystemEvents (Message, Facility,FromHost,  
Priority, DeviceReportedTime, ReceivedAt, InfoUnitID, SysLogTag) values ('%msg%',  
%syslogfacility%, '%HOSTNAME%',%syslogpriority%, '%timereported:::date-mysql%',  
'%timegenerated:::date-mysql%', %iut%, '%syslogtag%')",sql
```

rsyslog also contains a set of predefined templates identified by the **RSYSLOG_** prefix. These are reserved for the syslog use and it is advisable to not create a template using this prefix to avoid conflicts. The following list shows these predefined templates along with their definitions.

RSYSLOG_DebugFormat

A special format used for troubleshooting property problems

```
"Debug line with all properties:\nFROMHOST: '%FROMHOST%', fromhost-ip:  
'%fromhost-ip%', HOSTNAME: '%HOSTNAME%', PRI: %PRI%, \nsyslogtag  
'%syslogtag%', programname: '%programname%', APP-NAME: '%APP-NAME%', PROCID:  
'%PROCID%', MSGID: '%MSGID%', \nTIMESTAMP: '%TIMESTAMP%', STRUCTURED-DATA:
```

```
'%STRUCTURED-DATA%', \nmsg: '%msg%\nescaped msg: '%msg:::drop-cc%\nrawmsg:\nrawmsg'\n\n'
```

RSYSLOG_SyslogProtocol23Format

The format specified in IETF's internet-draft ietf-syslog-protocol-23, which is assumed to become the new syslog standard RFC.

```
"%PRI%1 %TIMESTAMP:::date-rfc3339% %HOSTNAME% %APP-NAME% %PROCID% %MSGID% %STRUCTURED-DATA% %msg%\n"
```

RSYSLOG_FileFormat

A modern-style logfile format similar to TraditionalFileFormat, but with high-precision time stamps and time zone information.

```
"%TIMESTAMP:::date-rfc3339% %HOSTNAME% %syslogtag% %msg:::sp-if-no-1st-sp% %msg:::drop-last-1f%\n"
```

RSYSLOG_TraditionalFileFormat

The older default log file format with low-precision time stamps.

```
"%TIMESTAMP% %HOSTNAME% %syslogtag% %msg:::sp-if-no-1st-sp% %msg:::drop-last-1f%\n"
```

RSYSLOG_ForwardFormat

A forwarding format with high-precision time stamps and time zone information.

```
"%PRI%%TIMESTAMP:::date-rfc3339% %HOSTNAME% %syslogtag:1:32% %msg:::sp-if-no-1st-sp% %msg%"
```

RSYSLOG_TraditionalForwardFormat

The traditional forwarding format with low-precision time stamps.

```
"%PRI%%TIMESTAMP% %HOSTNAME% %syslogtag:1:32% %msg:::sp-if-no-1st-sp% %msg%"
```

18.2.4. Global Directives

Global directives are configuration options that apply to the **rsyslogd** daemon. They usually specify a value for a specific predefined variable that affects the behavior of the **rsyslogd** daemon or a rule that follows. All of the global directives must start with a dollar sign (\$). Only one directive can be specified per line. The following is an example of a global directive that specifies the maximum size of the syslog message queue:

```
$MainMsgQueueSize 50000
```

The default size defined for this directive (**10,000** messages) can be overridden by specifying a different value (as shown in the example above).

You may define multiple directives in your `/etc/rsyslog.conf` configuration file. A directive affects the behavior of all configuration options until another occurrence of that same directive is detected. Global directives can be used to configure actions, queues and for debugging. A comprehensive list of all available configuration directives can be found in [Section 18.10, “Online Documentation”](#). Currently, a new configuration format has been developed that replaces the \$-based syntax (see [Section 18.2.6, “Using the New Configuration Format”](#)). However, classic global directives remain supported as a legacy format.

18.2.5. Log Rotation

The following is a sample `/etc/logrotate.conf` configuration file:

```
# rotate log files weekly
weekly
# keep 4 weeks worth of backlogs
rotate 4
# uncomment this if you want your log files compressed
compress
```

All of the lines in the sample configuration file define global options that apply to every log file. In our example, log files are rotated weekly, rotated log files are kept for four weeks, and all rotated log files are compressed by `gzip` into the `.gz` format. Any lines that begin with a hash sign (#) are comments and are not processed.

You may define configuration options for a specific log file and place it under the global options. However, it is advisable to create a separate configuration file for any specific log file in the `/etc/logrotate.d/` directory and define any configuration options there.

The following is an example of a configuration file placed in the `/etc/logrotate.d/` directory:

```
/var/log/messages {
    rotate 5
    weekly
    postrotate
        /usr/bin/killall -HUP syslogd
    endscript
}
```

The configuration options in this file are specific for the `/var/log/messages` log file only. The settings specified here override the global settings where possible. Thus the rotated `/var/log/messages` log file will be kept for five weeks instead of four weeks as was defined in the global options.

The following is a list of some of the directives you can specify in your `logrotate` configuration file:

- ▶ **`weekly`** — Specifies the rotation of log files to be done weekly. Similar directives include:
 - **`daily`**
 - **`monthly`**
 - **`yearly`**
- ▶ **`compress`** — Enables compression of rotated log files. Similar directives include:
 - **`nocompress`**
 - **`compresscmd`** — Specifies the command to be used for compressing.

- ***uncompresscmd***
- ***compressext*** — Specifies what extension is to be used for compressing.
- ***compressoptions*** — Lets you specify any options that may be passed to the used compression program.
- ***delaycompress*** — Postpones the compression of log files to the next rotation of log files.
- ▶ ***rotate INTEGER*** — Specifies the number of rotations a log file undergoes before it is removed or mailed to a specific address. If the value **0** is specified, old log files are removed instead of rotated.
- ▶ ***mail ADDRESS*** — This option enables mailing of log files that have been rotated as many times as is defined by the ***rotate*** directive to the specified address. Similar directives include:
 - ***nomail***
 - ***mailfirst*** — Specifies that the just-rotated log files are to be mailed, instead of the about-to-expire log files.
 - ***maillast*** — Specifies that the about-to-expire log files are to be mailed, instead of the just-rotated log files. This is the default option when ***mail*** is enabled.

For the full list of directives and various configuration options, see the **logrotate(5)** manual page.

18.2.6. Using the New Configuration Format

In **rsyslog** version 6, a new configuration syntax has been introduced. This new configuration format aims to be more powerful, more intuitive and to prevent common mistakes by not permitting certain invalid constructs. The syntax enhancement is enabled by the new configuration processor that relies on RainerScript. The legacy format is still fully supported and it is used by default in the **/etc/rsyslog.conf** configuration file.

RainerScript is a scripting language designed for processing network events and configuring event processors such as **rsyslog**. RainerScript was primarily used to define expression-based filters, see [Example 18.3, “Expression-based Filters”](#). The newest version of RainerScript implements the **input()** and **ruleset()** statements, which permit the **/etc/rsyslog.conf** configuration file to be written in new the style only.

In the following examples you can compare the configuration written with legacy-style parameters:

```
$InputFileName /tmp/inputfile
$InputFileTag tag1:
$InputFileStateFile inputfile-state
$InputRunFileMonitor
```

and the same configuration with use of the new format statement:

```
input(type="imfile" file="/tmp/inputfile" tag="tag1:" statefile="inputfile-state")
```

This significantly reduces the number of parameters used in configuration, improves readability, and also provides higher execution speed. For more information on RainerScript statements and parameters see [Section 18.10, “Online Documentation”](#).

18.2.7. Rulesets

Leaving special directives aside, **rsyslog** handles messages as defined by *rules* that consist of a filter condition and an action to be performed if the condition is true. With traditionally written **/etc/rsyslog.conf** file, all rules are evaluated in order of appearance for every input message. This process starts with the first rule and continues until all rules have been processed or until the message is discarded by one of the rules.

However, rules can be grouped into sequences called *rulesets*. With rulesets, you can limit the effect of certain rules only to selected inputs or enhance the performance of **rsyslog** by defining a distinct set of actions bound to a specific input. In other words, filter conditions that will be inevitably evaluated as false for certain types of messages can be skipped. With the new configuration format, the **input()** and **ruleset()** statements are reserved for this operation. The ruleset definition in **/etc/rsyslog.conf** can look as follows:

```
ruleset(name="rulesetname") {
    rule
    rule2
    call rulesetname2
    ...
}
```

Replace *rulesetname* with an identifier for your ruleset. The ruleset name cannot start with **RSYSLOG_** since this namespace is reserved for use by **rsyslog**. **RSYSLOG_DefaultRuleset** then defines the default set of rules to be performed if the message has no other ruleset assigned. With *rule* and *rule2* you can define rules in filter-action format mentioned above. With the **call** parameter, you can nest rulesets by calling them from inside other ruleset blocks.

After creating a ruleset, you need to specify what input it will apply to:

```
input(type="input_type" port="port_num" ruleset="rulesetname");
```

Here you can identify an input message by *input_type*, which is an input module that gathered the message, or by *port_num* – the port number. Other parameters such as *file* or *tag* can be specified for **input()**. Replace *rulesetname* with a name of the ruleset to be evaluated against the message. In case an input message is not explicitly bound to a ruleset, the default ruleset is triggered.

You can also use the legacy format to define rulesets, for more information see [Section 18.10, “Online Documentation”](#)

Example 18.11. Using rulesets

The following rulesets ensure different handling of remote messages coming from different ports. Add the following into `/etc/rsyslog.conf`:

```
ruleset(name="remote-10514") {
    action(type="omfile" file="/var/log/remote-10514")
}

ruleset(name="remote-10515") {
    cron.* action(type="omfile" file="/var/log/remote-10515-cron")
    mail.* action(type="omfile" file="/var/log/remote-10515-mail")
}

input(type="imtcp" port="10514" ruleset="remote-10514");
input(type="imtcp" port="10515" ruleset="remote-10515");
```

Rulesets shown in the above example define log destinations for the remote input from two ports, in case of 10515, messages are sorted according to the facility. Then, the TCP input is enabled and bound to rulesets. Note that you must load the required modules (imtcp) for this configuration to work.

18.2.8. Compatibility with syslogd

From **rsyslog** version 6, compatibility mode specified via the `-c` option has been removed. Also, the **syslogd**-style command line options are deprecated and configuring **rsyslog** through these command line options should be avoided. However, you can use several templates and directives to configure **rsyslogd** to emulate **syslogd**-like behavior.

For more information on various **rsyslogd** options, see the **rsyslogd(8)** manual page.

18.3. Working with Queues in Rsyslog

Queues are used to pass content, mostly syslog messages, between components of **rsyslog**. With queues, rsyslog is capable of processing multiple messages simultaneously and to apply several actions to a single message at once. The data flow inside **rsyslog** can be illustrated as follows:

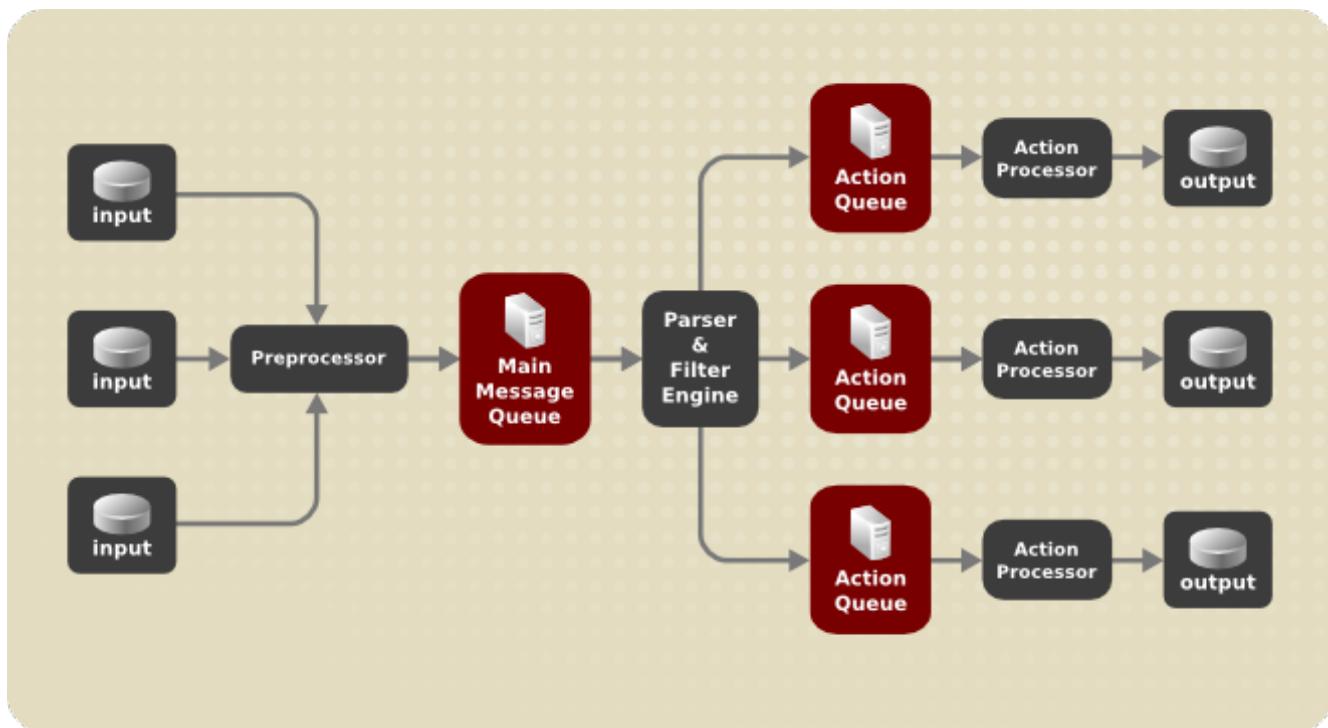


Figure 18.1. Message Flow in Rsyslog

Whenever **rsyslog** receives a message, it passes this message to the preprocessor and then places it into the *main message queue*. Messages wait there to be dequeued and passed to the *rule processor*.

The *rule processor* is a parsing and filtering engine. Here, the rules defined in `/etc/rsyslog.conf` are applied. Based on these rules, the rule processor evaluates which actions are to be performed. Each action has its own action queue. Messages are passed through this queue to the respective action processor which creates the final output. Note that at this point, several actions can run simultaneously on one message. For this purpose, a message is duplicated and passed to multiple action processors.

Only one queue per action is possible. Depending on configuration, the messages can be sent right to the action processor without action queuing. This is the behavior of *direct queues* (see below). In case the output action fails, the action processor notifies the action queue, which then takes an unprocessed element back and after some time interval, the action is attempted again.

To sum up, there are two positions where queues stand in **rsyslog**: either in front of the rule processor as a single *main message queue* or in front of various types of output actions as *action queues*. Queues provide two main advantages that both lead to increased performance of message processing:

- ▶ they serve as buffers that *decouple* producers and consumers in the structure of **rsyslog**
- ▶ they allow for *parallelization* of actions performed on messages

Apart from this, queues can be configured with several directives to provide optimal performance for your system. These configuration options are covered in the following chapters. For more information, see [Section 18.10. “Online Documentation”](#).

18.3.1. Defining Queues

Based on where the messages are stored, there are several types of queues: *direct*, *in-memory*, *disk*, and *disk-assisted in-memory* queues that are most widely used. You can choose one of these types for the main message queue and also for action queues. Add the following into `/etc/rsyslog.conf`:

```
$objectQueueType queue_type
```

Here, you can apply the setting for the main message queue (replace *object* with **MainMsg**) or for an action queue (replace *object* with **Action**). Replace *queue_type* with one of **direct**, **linkedlist** or **fixedarray** (which are in-memory queues), or **disk**.

The default setting for a main message queue is the **FixedArray** queue with a limit of 10,000 messages. Action queues are by default set as Direct queues.

Direct Queues

For many simple operations, such as when writing output to a local file, building a queue in front of an action is not needed. To avoid queuing, use:

```
$objectQueueType Direct
```

Replace *object* with **MainMsg** or with **Action** to use this option to the main message queue or for an action queue respectively. With direct queue, messages are passed directly and immediately from the producer to the consumer.

Disk Queues

Disk queues store messages strictly on hard drive, which makes them highly reliable but also the slowest of all possible queuing modes. This mode can be used to prevent the loss of highly important log data. However, disk queues are not recommended in most use cases. To set a disk queue, type the following into **/etc/rsyslog.conf**:

```
$objectQueueType Disk
```

Replace *object* with **MainMsg** or with **Action** to use this option to the main message queue or for an action queue respectively. Disk queues are written in parts of default size 10 Mb. This default size can be modified with the following configuration directive:

```
$objectQueueMaxFileSize size
```

where *size* represents the specified size of disk queue part. The defined size limit is not restrictive, **rsyslog** always writes one complete queue entry, even if it violates the size limit. Each part of a disk queue matches with an individual file. The naming directive for these files looks as follows:

```
$objectQueueFilename name
```

This sets a *name* prefix for the file followed by a 7-digit number starting at one and incremented for each file.

In-memory Queues

With in-memory queue, the enqueued messages are held in memory which makes the process very fast. The queued data are lost in case of a hard reset or after a shutdown. However, you can use the **\$ActionQueueSaveOnShutdown** setting to save the data before shutdown. There are two types of in-memory queues:

- ▶ *FixedSize* queue — the default mode for the main message queue, with a limit of 10,000 elements. This type of queue uses a fixed, pre-allocated array that holds pointers to queue elements. Due to these pointers, even if the queue is empty a certain amount of memory is consumed. However, *FixedSize* offers the best run time performance and is optimal when you expect a relatively low number of queued messages and high performance.

- ▶ *LinkedList* queue — here, all structures are dynamically allocated in a linked list, thus the memory is allocated only when needed. *LinkedList* queues handle occasional message bursts very well.

In general, use *LinkedList* queues when in doubt. Compared to *FixedArray*, it consumes less memory and lowers the processing overhead.

Use the following syntax to configure in-memory queues:

```
$objectQueueType LinkedList
```

```
$objectQueueType FixedArray
```

Replace *object* with **MainMsg** or with **Action** to use this option to the main message queue or for an action queue respectively.

Disk-Assisted In-memory Queues

Both disk and in-memory queues have their advantages and **rsyslog** lets you to combine them in *disk-assisted in-memory queues*. To do so, configure a normal in-memory queue and then add the *\$objectQueueFileName* directive to define a file name for disk assistance. This queue then becomes *disk-assisted*, which means it couples an in-memory queue with a disk queue to work in tandem.

The disk queue is activated if the in-memory queue is full or needs to persist after shutdown. With a disk-assisted queue, you can set both disk-specific and in-memory specific configuration parameters. This type of queue is probably the most commonly used, it is especially useful for potentially long-running and unreliable actions.

To specify the functioning of a disk-assisted in-memory queue, use the so-called *watermarks*:

```
$objectQueueHighWatermark number
```

```
$objectQueueLowWatermark number
```

Replace *object* with **MainMsg** or with **Action** to use this option to the main message queue or for an action queue respectively. Replace *number* with a number of enqueued messages. When an in-memory queue reaches the number defined by the high watermark, it starts writing messages to disk and continues until the in-memory queue size drops to the number defined with the low watermark. Correctly set watermarks minimize unnecessary disk writes, but also leave memory space for message bursts since writing to disk files is rather lengthy. Therefore, the high watermark must be lower than the whole queue capacity set with *\$objectQueueSize*. The difference between the high watermark and the overall queue size is a spare memory buffer reserved for message bursts. On the other hand, setting the high watermark too low will turn on disk assistance unnecessarily often.

18.3.2. Managing Queues

All types of queues can be further configured to match your requirements. You can use several directives to modify both action queues and the main message queue. Currently, there are more than 20 queue parameters available, see [Section 18.10, “Online Documentation”](#) Some of these settings are used commonly, others, such as worker thread management, provide closer control over the queue behavior and are reserved for advanced users. With advanced settings, you can optimize **rsyslog**'s performance, schedule queuing or modify the behavior of queue on system shutdown.

Limiting Queue Size

You can limit the number of messages that queue can contain with the following setting:

```
$objectQueueHighWatermark number
```

Replace *object* with **MainMsg** or with **Action** to use this option to the main message queue or for an action queue respectively. Replace *number* with a number of enqueued messages. You can set the queue size only as the number of messages, not as their actual memory size. The default queue size is 10,000 messages for the main message queue and ruleset queues, and 1000 for action queues.

Disk assisted queues are unlimited by default and can not be restricted with this directive, but you can reserve them physical disk space in bytes with the following settings:

```
$objectQueueMaxDiscSpace number
```

Replace *object* with **MainMsg** or with **Action**. When the size limit specified by *number* is hit, messages are discarded until sufficient amount of space is freed by dequeued messages.

Discarding Messages

When a queue reaches a certain number of messages, you can discard less important messages in order to save space in the queue for entries of higher priority. The threshold that launches the discarding process can be set with the so-called *discard mark*:

```
$objectQueueDiscardMark number
```

Replace *object* with **MainMsg** or with **Action** to use this option to the main message queue or for an action queue respectively. Here, *number* stands for a number of messages that have to be in the queue to start the discarding process. To define which messages to discard, use:

```
$objectQueueDiscardSeverity priority
```

Replace *priority* with one of the following keywords (or with a number): **debug** (7), **info** (6), **notice** (5), **warning** (4), **err** (3), **crit** (2), **alert** (1), and **emerg** (0). With this setting, both newly incoming and already queued messages with lower than defined priority are erased from the queue immediately after the discard mark is reached.

Using Timeframes

You can configure **rsyslog** to process queues during a specific time period. With this option you can, for example, transfer some processing into off-peak hours. To define a time frame, use the following syntax:

```
$objectQueueDequeueTimeBegin hour
```

```
$objectQueueDequeueTimeEnd hour
```

With *hour* you can specify hours that bound your time frame. Use the 24-hour format without minutes.

Configuring Worker Threads

A *worker thread* performs a specified action on the enqueued message. For example, in the main message queue, a worker task is to apply filter logic to each incoming message and enqueue them to the relevant action queues. When a message arrives, a worker thread is started automatically. When the number of messages reaches a certain number, another worker thread is turned on. To specify this number, use:

```
$objectQueueWorkerThreadMinimumMessages number
```

Replace *number* with a number of messages that will trigger a supplemental worker thread. For example, with *number* set to 100, a new worker thread is started when more than 100 messages arrive. When more than 200 messages arrive, the third worker thread starts and so on. However, too many working threads running in parallel becomes ineffective, so you can limit the maximum number of them by using:

```
$objectQueueWorkerThreads number
```

where *number* stands for a maximum number off working threads that can run in parallel. For the main message queue, the default limit is 1 thread. Once a working thread has been started, it keeps running until an inactivity timeout appears. To set the length of timeout, type:

```
$objectQueueWorkerTimeoutThreadShutdown time
```

Replace *time* with the duration set in milliseconds. Without this setting, a zero timeout is applied and a worker thread is terminated immediately when it runs out of messages. If you specify *time* as **-1**, no thread will be closed.

Batch Dequeuing

To increase performance, you can configure **rsyslog** to dequeue multiple messages at once. To set the upper limit for such dequeuing, use:

```
$objectQueueDequeueBatchSize number
```

Replace *number* with maximum number of messages that can be dequeued at once. Note that higher setting combined with higher number of permitted working threads results in greater memory consumption.

Terminating Queues

When terminating a queue that still contains messages, you can try to minimize the data loss by specifying a time interval for worker threads to finish the queue processing:

```
$objectQueueTimeoutShutdown time
```

Specify *time* in milliseconds. If after that period there are still some enqueued messages, workers finish the current data element and then terminate. Unprocessed messages are therefore lost. Another time interval can be set for workers to finish the final element:

```
$objectQueueTimeoutActionCompletion time
```

In case this timeout expires, any remaining workers are shut down. To save data at shutdown, use:

```
$objectQueueTimeoutSaveOnShutdown time
```

If set, all queue elements are saved to disk before **rsyslog** terminates.

18.4. Using Rsyslog Modules

Due to its modular design, **rsyslog** offers a variety of *modules* which provide additional functionality. Note that modules can be written by third parties. Most modules provide additional inputs (see *Input Modules* below) or outputs (see *Output Modules* below). Other modules provide special functionality specific to each module. The modules may provide additional configuration directives that become available after a module is loaded. To load a module, use the following syntax:

```
$ModLoad MODULE
```

where **\$ModLoad** is the global directive that loads the specified module and *MODULE* represents your desired module. For example, if you want to load the Text File Input Module (**imfile**) that enables **rsyslog** to convert any standard text files into syslog messages, specify the following line in the **/etc/rsyslog.conf** configuration file:

```
$ModLoad imfile
```

rsyslog offers a number of modules which are split into the following main categories:

- ▶ Input Modules — Input modules gather messages from various sources. The name of an input module always starts with the **im** prefix, such as **imfile**, **imjournal**, etc.
- ▶ Output Modules — Output modules provide a facility to issue message to various targets such as sending across a network, storing in a database, or encrypting. The name of an output module always starts with the **om** prefix, such as **omsnmp**, **omrelp**, etc.
- ▶ Parser Modules — These modules are useful in creating custom parsing rules or to parse malformed messages. With moderate knowledge of the C programming language, you can create your own message parser. The name of a parser module always starts with the **pm** prefix, such as **pmrfc5424**, **pmrfc3164**, and so on.
- ▶ Message Modification Modules — Message modification modules change content of syslog messages. Names of these modules start with the **mm** prefix. Message Modification Modules such as **mmanon**, **mmnormalize**, or **mmjsonparse** are used for anonymization or normalization of messages.
- ▶ String Generator Modules — String generator modules generate strings based on the message content and strongly cooperate with the template feature provided by **rsyslog**. For more information on templates, see [Section 18.2.3, “Templates”](#). The name of a string generator module always starts with the **sm** prefix, such as **smfile** or **smtradfile**.
- ▶ Library Modules — Library modules provide functionality for other loadable modules. These modules are loaded automatically by **rsyslog** when needed and cannot be configured by the user.

A comprehensive list of all available modules and their detailed description can be found at

http://www.rsyslog.com/doc/rsyslog_conf_modules.html



Warning

Note that when **rsyslog** loads any modules, it provides them with access to some of its functions and data. This poses a possible security threat. To minimize security risks, use trustworthy modules only.

18.4.1. Importing Text Files

The Text File Input Module, abbreviated as **imfile**, enables **rsyslog** to convert any text file into a stream of syslog messages. You can use **imfile** to import log messages from applications that create their own text file logs. To load **imfile**, add the following into **etc/rsyslog.conf**:

```
$ModLoad imfile
$InputFilePollInterval int
```

It is sufficient to load **imfile** once, even when importing multiple files. The **\$InputFilePollInterval** global

directive specifies how often **rsyslog** checks for changes in connected text files. The default interval is 10 seconds, to change it, replace *int* with a time interval specified in seconds.

To identify the text files to import, use the following syntax in **/etc/rsyslog.conf**:

```
# File 1
$InputFileName path_to_file
$InputFileTag tag:
$InputFileStateFile state_file_name
$InputFileSeverity severity
$InputFileFacility facility
$InputRunFileMonitor

# File 2
$InputFileName path_to_file2
...
```

Four settings are required to specify an input text file:

- ▶ replace *path_to_file* with a path to the text file
- ▶ replace *tag:* with a tag name for this message
- ▶ replace *state_file_name* with a unique name for the *state file*. *State files*, which are stored in the rsyslog working directory, keep cursors for the monitored files, marking what partition has already been processed. If you delete them, whole files will be read in again. Make sure that you specify a name that does not already exist.
- ▶ add the *\$InputRunFileMonitor* directive that enables the file monitoring. Without this setting, the text file will be ignored.

Apart from the required directives, there are several other settings that can be applied on the text input. Set the severity of messages by replacing *severity* or replace *facility* to define the subsystem that produced the message.

Example 18.12. Importing Text Files

The Apache HTTP server creates log files in text format. To apply the processing capabilities of **rsyslog** to apache error messages, first use the **imfile** module to import the messages. Add the following into **/etc/rsyslog.conf**:

```
$ModLoad imfile

$InputFileName /var/log/httpd/error_log
$InputFileTag apache-error:
$InputFileStateFile state-apache-error
$InputRunFileMonitor
```

18.4.2. Exporting Messages to a Database

Processing of log data can be faster and more convenient when performed in a database rather than with text files. Based on the type of DBMS used, chose from various output modules such as **ommysql**, **ompgsql**, **omoracle**, or **ommongodb**. As an alternative, use the generic **omlibdbi** output module that relies on the **libdbi** library. The **omlibdbi** module supports database systems Firebird/Interbase, MS SQL, Sybase, SQLite, Ingres, Oracle, mSQL, MySQL, and PostgreSQL.

Example 18.13. Exporting Rsyslog Messages to a Database

To store the rsyslog messages in a MySQL database, add the following into `/etc/rsyslog.conf`:

```
$ModLoad ommysql
$ActionOmmysqlServerPort 1234
*.*:ommysql:database-server, database-name, database-userid, database-password
```

First, the output module is loaded, then the communication port is specified. Additional information, such as name of the server and the database, and authentication data, is specified on the last line of the above example.

18.4.3. Enabling Encrypted Transport

Confidentiality and integrity in network transmissions can be provided by either the *TLS* or *GSSAPI* encryption protocol.

Transport Layer Security (*TLS*) is a cryptographic protocol designed to provide communication security over the network. When using *TLS*, rsyslog messages are encrypted before sending, and mutual authentication exists between the sender and receiver.

Generic Security Service API (*GSSAPI*) is an application programming interface for programs to access security services. To use it in connection with **rsyslog** you must have a functioning **Kerberos** environment.

18.4.4. Using RELP

Reliable Event Logging Protocol (*RELP*) is a networking protocol for data logging in computer networks. It is designed to provide a reliable delivery of event messages, which makes it useful in environments where message loss is not acceptable.

18.5. Interaction of Rsyslog and Journal

As mentioned above, **Rsyslog** and **Journal**, the two logging applications present on your system, have several distinctive features that make them suitable for specific use cases. In many situations it is useful to combine their capabilities, for example to create structured messages and store them in a file database (see [Section 18.6, “Structured Logging with Rsyslog”](#)). A communication interface needed for this cooperation is provided by input and output modules on the side of **Rsyslog** and by the **Journal**'s communication socket.

By default, **rsyslogd** uses the **imjournal** module as a default input mode for journal files. With this module, you import not only the messages but also the structured data provided by **journald**. Also, older data can be imported from **journald** (unless forbidden with the **\$ImjournalIgnorePreviousMessages** directive). See [Section 18.6.1, “Importing Data from Journal”](#) for basic configuration of **imjournal**.

As an alternative, configure **rsyslogd** to read from the socket provided by **journald** as an output for syslog-based applications. The path to the socket is `/run/systemd/journal/syslog`. Use this option when you wish to maintain plain rsyslog messages. Compared to **imjournal** the socket input currently offers more features, such as ruleset binding or filtering. To import **Journal** data through the socket, use the following configuration in `/etc/rsyslog.conf`:

```
$ModLoad imuxsock
$OmitLocalLogging off
```

The above syntax loads the **imuxsock** module and turns off the **\$OmitLocalLogging** directive, which enables the import through the system socket. The path to this socket is specified separately in **/etc/rsyslog.d/listen.conf** as follows:

```
$SystemLogSocketName /run/systemd/journal/syslog
```

You can also output messages from **Rsyslog** to **Journal** with the **omjournal** module. Configure the output in **/etc/rsyslog.conf** as follows:

```
$ModLoad omjournal
*.* :omjournal:
```

For instance, the following configuration forwards all received on tcp port 10514 to the Journal:

```
$ModLoad imtcp
$ModLoad omjournal

$RuleSet remote
*.* :omjournal:

$InputTCPServerBindRuleset remote
$InputTCPServerRun 10514
```

18.6. Structured Logging with Rsyslog

On systems that produce large amounts of log data, it can be convenient to maintain log messages in a *structured format*. With structured messages, it is easier to search for particular information, to produce statistics and to cope with changes and inconsistencies in message structure. **Rsyslog** uses the **JSON** (JavaScript Object Notation) format to provide structure for log messages.

Compare the following unstructured log message:

```
Oct 25 10:20:37 localhost anacron[1395]: Jobs will be executed sequentially
```

with a structured one:

```
{"timestamp":"2013-10-25T10:20:37", "host":"localhost", "program":"anacron",
"pid":1395, "msg":"Jobs will be executed sequentially"}
```

Searching structured data with use of key-value pairs is faster and more precise than searching text files with regular expressions. The structure also lets you to search for the same entry in messages produced by various applications. Also, JSON files can be stored in a document database such as MongoDB, which provides additional performance and analysis capabilities. On the other hand, a structured message requires more disk space than the unstructured one.

In **rsyslog**, log messages with meta data are pulled from **Journal** with use of the **imjournal** module. With the **mmjsonparse** module, you can parse data imported from **Journal** and form other sources and process them further, for example as a database output. For parsing to be successful, **mmjsonparse** requires input messages to be structured in a way that is defined by the **Lumberjack** project.

The **Lumberjack** project aims to add structured logging to **rsyslog** in a backward-compatible way. To identify a structured message, **Lumberjack** specifies the **@cee:** string that prepends the actual JSON structure. Also, **Lumberjack** defines the list of standard field names that should be used for entities in the JSON string. For more information on **Lumberjack**, see [Section 18.10, “Online Documentation”](#).

The following is an example of a lumberjack-formatted message:

```
@cee: {"pid":17055, "uid":1000, "gid":1000, "appname":"logger", "msg":"Message text."}
```

To build this structure inside **Rsyslog**, a template is used, see [Section 18.6.2, “Filtering Structured Messages”](#). Applications and servers can employ the **libumberlog** library to generate messages in the lumberjack-compliant form. For more information on **libumberlog**, see [Section 18.10, “Online Documentation”](#).

18.6.1. Importing Data from Journal

The **imjournal** module is **Rsyslog**'s input module to natively read the journal files (see [Section 18.5, “Interaction of Rsyslog and Journal”](#)). Journal messages are then logged in text format as other rsyslog messages. However, with further processing, it is possible to translate meta data provided by **Journal** into a structured message.

To import data from **Journal** to **Rsyslog**, use the following configuration in **/etc/rsyslog.conf**:

```
$ModLoad imjournal
$imjournalPersistStateInterval number_of_messages
$imjournalStateFile path
$imjournalRateLimitInterval seconds
$imjournalRateLimitBurst burst_number
$ImjournalIgnorePreviousMessages off/on
```

- ▶ With *number_of_messages*, you can specify how often the journal data must be saved. This will happen each time the specified number of messages is reached.
- ▶ Replace *path* with a path to the state file. This file tracks the journal entry that was the last one processed.
- ▶ With *seconds*, you set the length of the rate limit interval. The number of messages processed during this interval can not exceed the value specified in *burst_number*. The default setting is 20,000 messages per 600 seconds. Rsyslog discards messages that come after the maximum burst within the time frame specified.
- ▶ With **\$ImjournalIgnorePreviousMessages** you can ignore messages that are currently in Journal and import only new messages, which is used when there is no state file specified. The default setting is **off**. Please note that if this setting is off and there is no state file, all messages in the Journal are processed, even if they were already processed in a previous rsyslog session.



Note

You can use **imjournal** simultaneously with **imuxsock** module that is the traditional system log input. However, to avoid message duplication, you must prevent **imuxsock** from reading the Journal's system socket. To do so, use the **\$OmitLocalLogging** directive:

```
$ModLoad imuxsock
$ModLoad imjournal

$OmitLocalLogging on
$AddUnixListenSocket /run/systemd/journal/syslog
```

You can translate all data and meta data stored by **Journal** into structured messages. Some of these meta data entries are listed in [Example 18.15, “Verbose journalctl Output”](#), for a complete list of journal fields see the **systemd.journal-fields(7)** manual page. For example, it is possible to focus on *kernel journal fields*, that are used by messages originating in the kernel.

18.6.2. Filtering Structured Messages

To create a lumberjack-formatted message that is required by **rsyslog**'s parsing module, use the following template:

```
template(name="CEETemplate" type="string" string="%TIMESTAMP% %HOSTNAME% %syslogtag%
@cee: %%!all-json%\n")
```

This template prepends the **@cee:** string to the JSON string and can be applied, for example, when creating an output file with **omfile** module. To access JSON field names, use the **\$!** prefix. For example, the following filter condition searches for messages with specific *hostname* and *UID*:

```
($!hostname == "hostname" && $!UID== "UID")
```

18.6.3. Parsing JSON

The **mmjsonparse** module is used for parsing structured messages. These messages can come from **Journal** or from other input sources, and must be formatted in a way defined by the **Lumberjack** project. These messages are identified by the presence of the **@cee:** string. Then, **mmjsonparse** checks if the JSON structure is valid and then the message is parsed.

To parse lumberjack-formatted JSON messages with **mmjsonparse**, use the following configuration in the **/etc/rsyslog.conf**:

```
$ModLoad mmjsonparse
*.* :mmjsonparse:
```

In this example, the **mmjsonparse** module is loaded on the first line, then all messages are forwarded to it. Currently, there are no configuration parameters available for **mmjsonparse**.

18.6.4. Storing Messages in the MongoDB

Rsyslog supports storing JSON logs in the MongoDB document database through the **ommongodb** output module.

To forward log messages into MongoDB, use the following syntax in the `/etc/rsyslog.conf` (configuration parameters for `omongodb` are available only in the new configuration format; see [Section 18.2.6, “Using the New Configuration Format”](#)):

```
$ModLoad mongodb
*.* action(type="omongodb" server="DB_server" serverport="port" db="DB_name" collection="collection_name" uid="UID" pwd="password")
```

- ▶ Replace *DB_server* with the name or address of the MongoDB server. Specify *port* to select a non-standard port from the MongoDB server. The default *port* value is **0** and usually there is no need to change this parameter.
- ▶ With *DB_name*, you identify to which database on the MongoDB server you want to direct the output. Replace *collection_name* with the name of a collection in this database. In MongoDB, collection is a group of documents, the equivalent of an RDBMS table.
- ▶ You can set your login details by replacing *UID* and *password*.

You can shape the form of the final database output with use of templates. By default, **rsyslog** uses a template based on standard **lumberjack** field names.

18.7. Debugging Rsyslog

To run **rsyslogd** in debugging mode, use the following command:

```
rsyslogd -dn
```

With this command, **rsyslogd** produces debugging information and prints it to the standard output. The **-n** stands for "no fork". You can modify debugging with environmental variables, for example, you can store the debug output in a log file. Before starting **rsyslogd**, type the following on the command line:

```
export RSYSLOG_DEBUGLOG="path"
export RSYSLOG_DEBUG="Debug"
```

Replace *path* with a desired location for the file where the debugging information will be logged. For a complete list of options available for the RSYSLOG_DEBUG variable, see the related section in the **rsyslogd(8)** manual page.

To check if syntax used in the `etc/rsyslog.conf` file is valid use:

```
rsyslogd -N 1
```

Where **1** represents level of verbosity of the output message. This is a forward compatibility option because currently, only one level is provided. However, you must add this argument to run the validation.

18.8. Using the Journal

The Journal is a component of **systemd** that is responsible for viewing and management of log files. It can be used in parallel, or in place of a traditional syslog daemon, such as **rsyslogd**. The Journal was developed to address problems connected with traditional logging. It is closely integrated with the rest of the system, supports various logging technologies and access management for the log files.

Logging data are collected, stored, and processed by the Journal's **journald** service. It creates and maintains binary files called *journals* based on logging information that is received from the kernel, from user processes, from standard output, and standard error output of system services or via its native API. These journals are structured and indexed, which provides relatively fast seek times. Journal entries can carry a unique identifier. The **journald** service collects numerous meta data fields for each log message and the actual journal files are secured.

18.8.1. Viewing Log Files

To access the journal logs, use the **journalctl** tool. For a basic view of the logs type as **root**:

```
journalctl
```

An output of this command is a list of all log files generated on the system including messages generated by system components and by users. The structure of this output is similar to one used in **/var/log/messages/** but with certain improvements:

- ▶ the priority of entries is marked visually. Lines of error priority and higher are highlighted with red color and a bold font is used for lines with notice and warning priority
- ▶ the time stamps are converted for the local time zone of your system
- ▶ all logged data are shown, including rotated logs
- ▶ the beginning of a boot is tagged with a special line

Example 18.14. Example Output of journalctl

The following is an example output provided by the **journalctl** tool. When called without parameters, the listed entries begin with a time stamp, then the host name and application that performed the operation is mentioned followed by the actual message. This example shows the first three entries in the journal log:

```
# journalctl
-- Logs begin at Thu 2013-08-01 15:42:12 CEST, end at Thu 2013-08-01 15:48:48
CEST. --
Aug 01 15:42:12 localhost systemd-journal[54]: Allowing runtime journal files to
grow to 49.7M.
Aug 01 15:42:12 localhost kernel: Initializing cgroup subsys cpuset
Aug 01 15:42:12 localhost kernel: Initializing cgroup subsys cpu
[...]
```

In many cases, only the latest entries in the journal log are relevant. The simplest way to reduce **journalctl** output is to use the **-n** option that lists only the specified number of most recent log entries:

```
journalctl -n Number
```

Replace *Number* with the number of lines to be shown. When no number is specified, **journalctl** displays the ten most recent entries.

The **journalctl** command allows controlling the form of the output with the following syntax:

```
journalctl -o form
```

Replace *form* with a keyword specifying a desired form of output. There are several options, such as **verbose**, which returns full-structured entry items with all fields, **export**, which creates a binary stream suitable for backups and network transfer, and **json**, which formats entries as JSON data structures. For the full list of keywords, see the **journalctl(1)** manual page.

Example 18.15. Verbose journalctl Output

To view full meta data about all entries, type:

```
# journalctl -o verbose
[...]
Fri 2013-08-02 14:41:22 CEST
[s=e1021ca1b81e4fc688fad6a3ea21d35b;i=55c;b=78c81449c920439da57da7bd5c56a770;m=27
cc
    _BOOT_ID=78c81449c920439da57da7bd5c56a770
    PRIORITY=5
    SYSLOG_FACILITY=3
    _TRANSPORT=syslog
    _MACHINE_ID=69d27b356a94476da859461d3a3bc6fd
    _HOSTNAME=localhost.localdomain
    _PID=562
    _COMM=dbus-daemon
    _EXE=/usr/bin/dbus-daemon
    _CMDLINE=/bin/dbus-daemon --system --address=systemd: --nofork --
nopidfile --systemd-activation
    _SYSTEMD_CGROUP=/system/dbus.service
    _SYSTEMD_UNIT=dbus.service
    SYSLOG_IDENTIFIER=dbus
    SYSLOG_PID=562
    _UID=81
    _GID=81
    _SELINUX_CONTEXT=system_u:system_r:system_dbusd_t:s0-s0:c0.c1023
    MESSAGE=[system] Successfully activated service 'net.reactivated.Fprint'
    _SOURCE_REALTIME_TIMESTAMP=1375447282839181
[...]
```

This example lists fields that identify a single log entry. These meta data can be used for message filtering as shown in [Section 18.8.4, “Advanced Filtering”](#). For a complete description of all possible fields see the **systemd.journal-fields(7)** manual page.

18.8.2. Access Control

By default, **Journal** users without **root** privileges can only see log files generated by them. The system administrator can add selected users to the **adm** group, which grants them access to complete log files. To do so, type as **root**:

```
usermod -a -G adm username
```

Here, replace *username* with a name of the user to be added to the **adm** group. This user then receives the same output of the **journalctl** command as the root user. Note that access control only works when persistent storage is enabled for **Journal**.

18.8.3. Using The Live View

When called without parameters, **journalctl** shows the full list of entries, starting with the oldest entry collected. With the live view, you can supervise the log messages in real time as new entries are continuously printed as they appear. To start **journalctl** in live view mode, type:

```
journalctl -f
```

This command returns a list of the ten most current log lines. The **journalctl** utility then stays running and waits for new changes to show them immediately.

18.8.4. Filtering Messages

The output of the **journalctl** command executed without parameters is often extensive, therefore you can use various filtering methods to extract information to meet your needs.

Filtering by Priority

Log messages are often used to track erroneous behavior on the system. To view only entries with a selected or higher priority, use the following syntax:

```
journalctl -p priority
```

Here, replace *priority* with one of the following keywords (or with a number): **debug** (7), **info** (6), **notice** (5), **warning** (4), **err** (3), **crit** (2), **alert** (1), and **emerg** (0).

Example 18.16. Filtering by Priority

To view only entries with *error* or higher priority, use:

```
journalctl -p err
```

Filtering by Time

To view log entries only from the current boot, type:

```
journalctl -b
```

If you reboot your system just occasionally, the **-b** will not significantly reduce the output of **journalctl**. In such cases, time-based filtering is more helpful:

```
journalctl --since=value --until=value
```

With **--since** and **--until**, you can view only log messages created within a specified time range. You can pass *values* to these options in form of date or time or both as shown in the following example.

Example 18.17. Filtering by Time and Priority

Filtering options can be combined to narrow the set of results according to specific request requests. For example, to view the *warning* or higher priority messages from certain point in time, type:

```
journalctl -p warning --since="2013-3-16 23:59:59"
```

Advanced Filtering

[Example 18.15, “Verbose journalctl Output”](#) lists a set of fields that specify a log entry and can all be used for filtering. For a complete description of meta data that **systemd** can store, see the **systemd.journal-fields(7)** manual page. This meta data is collected for each log message, without user intervention. Values are usually text-based, but can take binary and large values; fields can have multiple values assigned though it is not very common.

To view a list of unique values that occur in a specified field, use the following syntax:

```
journalctl -F fieldname
```

Replace *fieldname* with a name of a field you are interested in.

To show only log entries that fit a specific condition, use the following syntax:

```
journalctl fieldname=value
```

Replace *fieldname* with a name of a field and *value* with a specific value contained in that field. As a result, only lines that match this condition are returned.



Tab Completion on Field Names

As the number of meta data fields stored by **systemd** is quite large, it is easy to forget the exact name of the field of interest. When unsure, type:

```
journalctl
```

and press the **Tab** key two times. This shows a list of available field names. **Tab** completion based on context works on field names, so you can type a distinctive set of letters from a field name and then press **Tab** to complete the name automatically. Similarly, you can list unique values from a field. Type:

```
journalctl fieldname=
```

and press **Tab** two times. This serves as an alternative to **journalctl -F fieldname**.

You can specify multiple values for one field:

```
journalctl fieldname=value1 fieldname=value2 ...
```

Specifying two matches for the same field results in a *logical OR* combination of the matches. Entries matching *value1* or *value2* are displayed.

Also, you can specify multiple field-value pairs to further reduce the output set:

```
journalctl fieldname1=value1 fieldname2=value2 ...
```

If two matches for different field names are specified, they will be combined with a *logical AND*. Entries have to match both conditions to be shown.

With use of the + symbol, you can set a *logical OR* combination of matches for multiple fields:

```
journalctl fieldname1=value + fieldname2=value ...
```

This command returns entries that match at least one of the conditions, not only those that match the both of them.

Example 18.18. Advanced filtering

To display entries created by **avahi-daemon.service** or **crond.service** under user with UID 70, use the following command:

```
journalctl _UID=70 _SYSTEMD_UNIT=avahi-daemon.service  
_SYSTEMD_UNIT=crond.service
```

Since there are two values set for the **_SYSTEMD_UNIT** field, both results will be displayed, but only when matching the **_UID=70** condition. This can be expressed simply as: (UID=70 and (avahi or cron)).

You can apply the aforementioned filtering also in the live-view mode to keep track of the latest changes in the selected group of log entries:

```
journalctl -f fieldname=value ...
```

18.8.5. Enabling Persistent Storage

By default, **Journal** stores log files only in memory or a small ring-buffer in the **/run/log/journal/** directory. This is sufficient to show recent log history with **journalctl**. This directory is volatile, log data are not saved permanently. With the default configuration, syslog reads the journal logs and stores them in the **/var/log/** directory. With persistent logging enabled, journal files are stored in **/var/log/journal** which means they persist after reboot. Journal can then replace **rsyslog** for some users (but see the chapter introduction).

Enabled persistent storage has the following advantages

- ▶ Richer data is recorded for troubleshooting in a longer period of time
- ▶ For immediate troubleshooting, richer data is available after a reboot
- ▶ Server console currently reads data from journal, not log files

Persistent storage has also certain disadvantages:

- ▶ Even with persistent storage the amount of data stored depends on free memory, there is no guarantee to cover a specific time span
- ▶ More disk space is needed for logs

To enable persistent storage for Journal, create the journal directory manually as shown in the following example. As **root** type:

```
mkdir -p /var/log/journal
```

Then, restart **journald** to apply the change:

```
systemctl restart systemd-journald
```

18.9. Managing Log Files in Graphical Environment

18.9.1. Viewing Log Files

Most log files are stored in plain text format. You can view them with any text editor such as **Vi** or **Emacs**. Some log files are readable by all users on the system; however, root privileges are required to read most log files.

To view system log files in an interactive, real-time application, use the **Log File Viewer**.

Installing the *gnome-system-log* package

In order to use the **Log File Viewer**, first ensure the *gnome-system-log* package is installed on your system by running, as **root**:

```
~]# yum install gnome-system-log
```

For more information on installing packages with Yum, see [Section 5.2.4, “Installing Packages”](#).

After you have installed the *gnome-system-log* package, open the **Log File Viewer** by clicking **Applications** → **System Tools** → **Log File Viewer**, or type the following command at a shell prompt:

```
~]$ gnome-system-log
```

The application only displays log files that exist; thus, the list might differ from the one shown in [Figure 18.2, “Log File Viewer”](#).

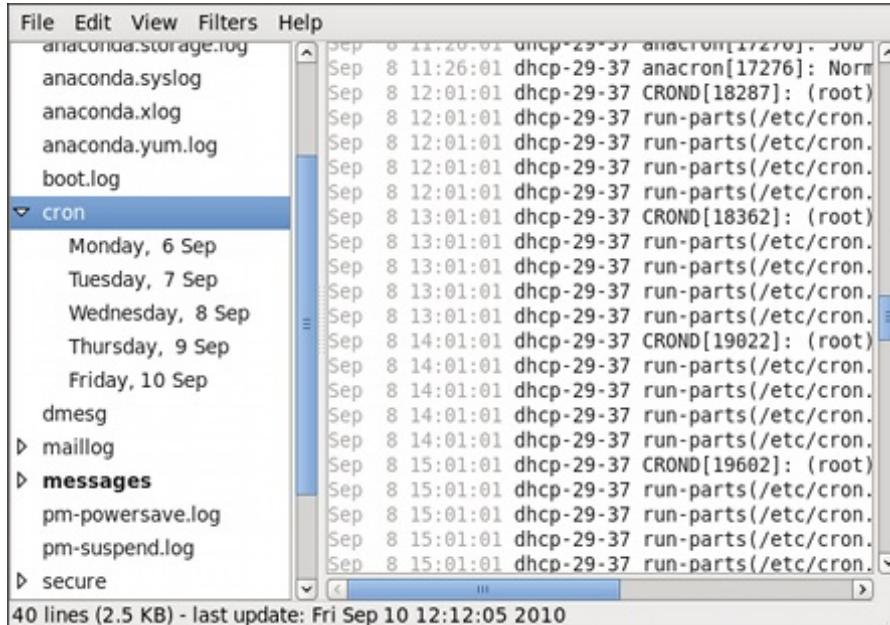


Figure 18.2. Log File Viewer

The **Log File Viewer** application lets you filter any existing log file. Click on **Filters** from the menu and select **Manage Filters** to define or edit the desired filter.

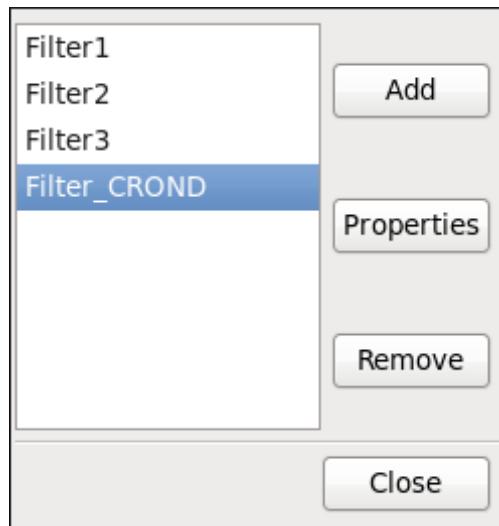


Figure 18.3. Log File Viewer - Filters

Adding or editing a filter lets you define its parameters as is shown in [Figure 18.4, “Log File Viewer - defining a filter”](#).

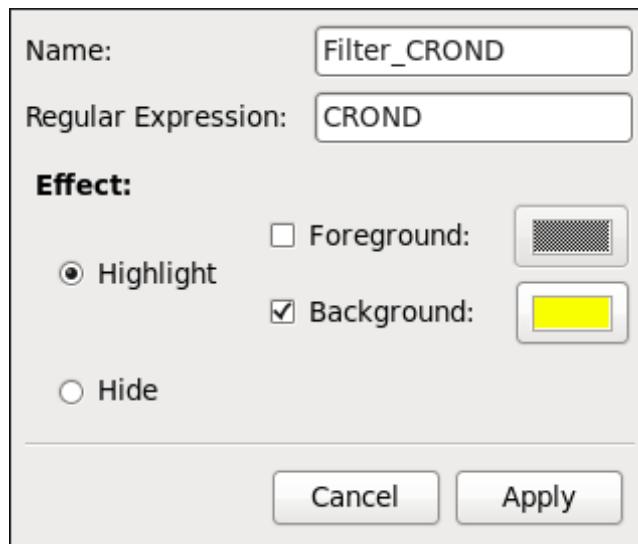


Figure 18.4. Log File Viewer - defining a filter

When defining a filter, the following parameters can be edited:

- ▶ **Name** — Specifies the name of the filter.
- ▶ **Regular Expression** — Specifies the regular expression that will be applied to the log file and will attempt to match any possible strings of text in it.
- ▶ **Effect**
 - **Highlight** — If checked, the found results will be highlighted with the selected color. You may select whether to highlight the background or the foreground of the text.
 - **Hide** — If checked, the found results will be hidden from the log file you are viewing.

When you have at least one filter defined, it can be selected from the **Filters** menu and it will automatically search for the strings you have defined in the filter and highlight or hide every successful match in the log file you are currently viewing.

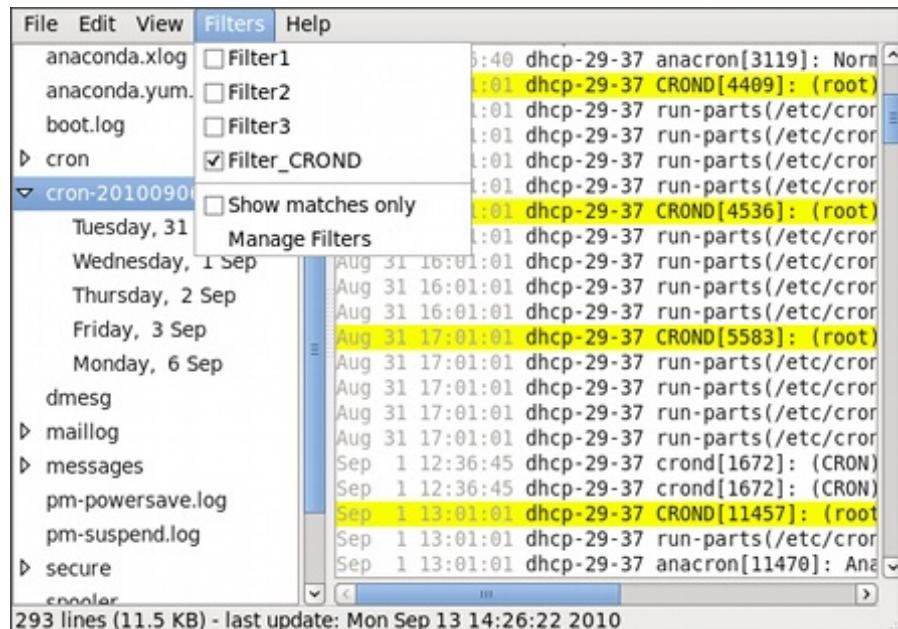


Figure 18.5. Log File Viewer - enabling a filter

When you select the **Show matches only** option, only the matched strings will be shown in the log file you are currently viewing.

18.9.2. Adding a Log File

To add a log file you wish to view in the list, select **File → Open**. This will display the **Open Log** window where you can select the directory and file name of the log file you wish to view. [Figure 18.6, “Log File Viewer - adding a log file”](#) illustrates the **Open Log** window.

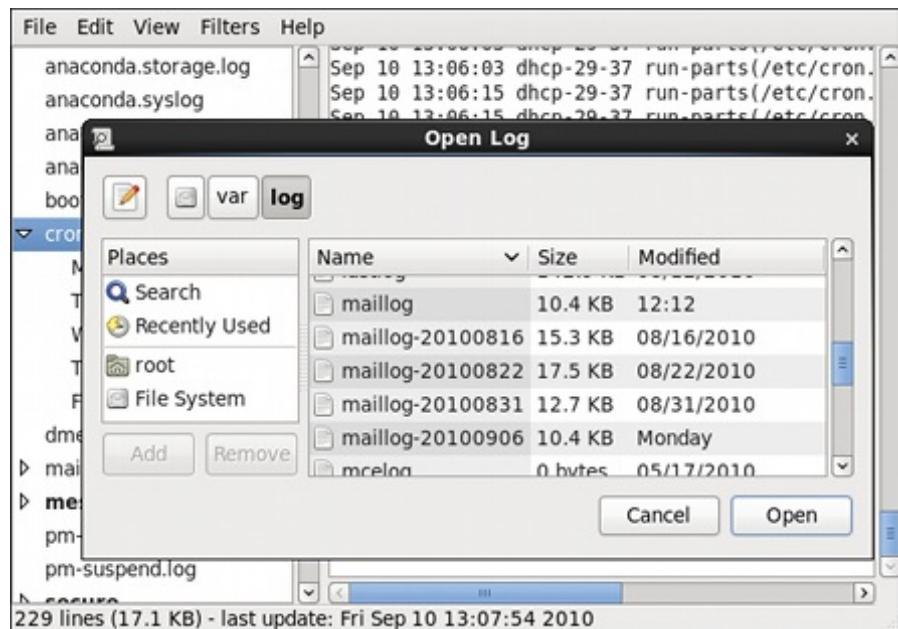


Figure 18.6. Log File Viewer - adding a log file

Click on the **Open** button to open the file. The file is immediately added to the viewing list where you can select it and view its contents.



Reading zipped log files

The **Log File Viewer** also allows you to open log files zipped in the **.gz** format.

18.9.3. Monitoring Log Files

Log File Viewer monitors all opened logs by default. If a new line is added to a monitored log file, the log name appears in bold in the log list. If the log file is selected or displayed, the new lines appear in bold at the bottom of the log file. [Figure 18.7, “Log File Viewer - new log alert”](#) illustrates a new alert in the **cron** log file and in the **messages** log file. Clicking on the **cron** log file displays the logs in the file with the new lines in bold.

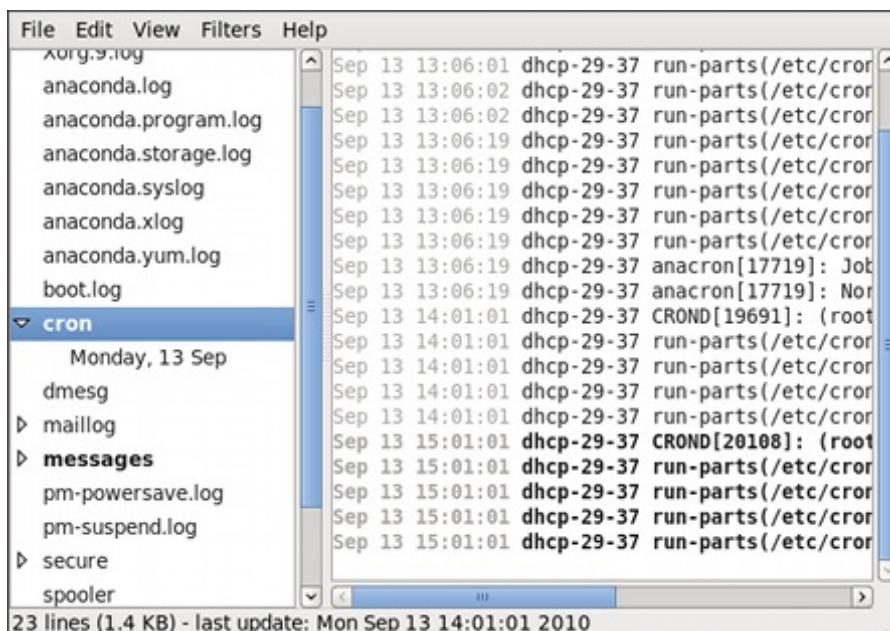


Figure 18.7. Log File Viewer - new log alert

18.10. Additional Resources

For more information on how to configure the **rsyslog** daemon and how to locate, view, and monitor log files, see the resources listed below.

Installed Documentation

- ▶ **rsyslogd(8)** — The manual page for the **rsyslogd** daemon documents its usage.
- ▶ **rsyslog.conf(5)** — The manual page named **rsyslog.conf** documents available configuration options.
- ▶ **logrotate(8)** — The manual page for the **logrotate** utility explains in greater detail how to configure and use it.
- ▶ **journalctl(1)** — The manual page for the **journalctl** daemon documents its usage.
- ▶ **journald.conf(5)** — This manual page documents available configuration options.
- ▶ **systemd.journal-fields(7)** — This manual page lists special **Journal** fields.

Online Documentation

- ▶ [rsyslog Home Page](#) — The **rsyslog** home page offers a thorough technical breakdown of its features, documentation, configuration examples, and video tutorials.
- ▶ [RainerScript documentation on the rsyslog Home Page](#) — Commented summary of data types, expressions, and functions available in *RainerScript*.
- ▶ [Description of queues on the rsyslog Home Page](#) — General information on various types of message queues and their usage.
- ▶ [rsyslog Wiki](#) — The *rsyslog* *Wiki* contains useful configuration examples.

See Also

- ▶ [Chapter 4, Gaining Privileges](#) documents how to gain administrative privileges by using the **su** and **sudo** commands.
- ▶ [Chapter 6, Managing Services with systemd](#) provides more information on **systemd** and documents how to use the **systemctl** command to manage system services.

Chapter 19. Automating System Tasks

Tasks, also known as *jobs*, can be configured to run automatically within a specified period of time, on a specified date, or when the system load average decreases below 0.8.

Red Hat Enterprise Linux is pre-configured to run important system tasks to keep the system updated. For example, the slocate database used by the **locate** command is updated daily. A system administrator can use automated tasks to perform periodic backups, monitor the system, run custom scripts, and so on.

Red Hat Enterprise Linux comes with the following automated task utilities: **cron**, **anacron**, **at**, and **batch**.

Every utility is intended for scheduling a different job type: while Cron and Anacron schedule recurring jobs, At and Batch schedule one-time jobs (refer to [Section 19.1, “Cron and Anacron”](#) and [Section 19.2, “At and Batch”](#) respectively).

Red Hat Enterprise Linux 7 supports the use of **systemd.timer** for executing a job at a specific time. See man **systemd.timer(5)** for more information.

19.1. Cron and Anacron

Both Cron and Anacron are daemons that can schedule execution of recurring tasks to a certain point in time defined by the exact time, day of the month, month, day of the week, and week.

Cron jobs can run as often as every minute. However, the utility assumes that the system is running continuously and if the system is not on at the time when a job is scheduled, the job is not executed.

On the other hand, Anacron remembers the scheduled jobs if the system is not running at the time when the job is scheduled. The job is then executed as soon as the system is up. However, Anacron can only run a job once a day.

19.1.1. Installing Cron and Anacron

To install Cron and Anacron, you need to install the *cronie* package with Cron and the *cronie-anacron* package with Anacron (*cronie-anacron* is a sub-package of *cronie*).

To determine if the packages are already installed on your system, issue the following command:

```
rpm -q cronie cronie-anacron
```

The command returns full names of the *cronie* and *cronie-anacron* packages if already installed, or notifies you that the packages are not available.

To install these packages, use the **yum** command in the following form as **root**:

```
yum install package
```

For example, to install both Cron and Anacron, type the following at a shell prompt:

```
~]# yum install cronie cronie-anacron
```

For more information on how to install new packages in Red Hat Enterprise Linux, see [Section 5.2.4, “Installing Packages”](#).

19.1.2. Running the Crond Service

The cron and anacron jobs are both picked by the **crond** service. This section provides information on how to start, stop, and restart the **crond** service, and shows how to configure it to start automatically at boot time. For more information on how to manage system service in Red Hat Enterprise Linux 7 in general, see [Chapter 6, Managing Services with systemd](#).

19.1.2.1. Starting and Stopping the Cron Service

To determine if the service is running, use the following command:

```
systemctl status crond.service
```

To run the **crond** service in the current session, type the following at a shell prompt as **root**:

```
systemctl start crond.service
```

To configure the service to start automatically at boot time, use the following command as **root**:

```
systemctl enable crond.service
```

19.1.2.2. Stopping the Cron Service

To stop the **crond** service in the current session, type the following at a shell prompt as **root**:

```
systemctl stop crond.service
```

To prevent the service from starting automatically at boot time, use the following command as **root**:

```
systemctl disable crond.service
```

19.1.2.3. Restarting the Cron Service

To restart the **crond** service, type the following at a shell prompt as **root**:

```
systemctl restart crond.service
```

This command stops the service and starts it again in quick succession.

19.1.3. Configuring Anacron Jobs

The main configuration file to schedule jobs is the **/etc/anacrontab** file, which can be only accessed by the **root** user. The file contains the following:

```
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=45
# the jobs will be started during the following hours only
START_HOURS_RANGE=3-22

#period in days    delay in minutes    job-identifier    command
1              5      cron.daily      nice run-parts /etc/cron.daily
7              25     cron.weekly     nice run-parts /etc/cron.weekly
@monthly      45     cron.monthly   nice run-parts /etc/cron.monthly
```

The first three lines define the variables that configure the environment in which the anacron tasks run:

- ▶ **SHELL** — shell environment used for running jobs (in the example, the Bash shell)
- ▶ **PATH** — paths to executable programs
- ▶ **MAILTO** — username of the user who receives the output of the anacron jobs by email

If the **MAILTO** variable is not defined (**MAILTO=**), the email is not sent.

The next two variables modify the scheduled time for the defined jobs:

- ▶ **RANDOM_DELAY** — maximum number of minutes that will be added to the **delay in minutes** variable which is specified for each job

The minimum delay value is set, by default, to 6 minutes.

If **RANDOM_DELAY** is, for example, set to **12**, then between 6 and 12 minutes are added to the **delay in minutes** for each job in that particular anacrontab. **RANDOM_DELAY** can also be set to a value below **6**, including **0**. When set to **0**, no random delay is added. This proves to be useful when, for example, more computers that share one network connection need to download the same data every day.

- ▶ **START_HOURS_RANGE** — interval, when scheduled jobs can be run, in hours

In case the time interval is missed, for example due to a power failure, the scheduled jobs are not executed that day.

The remaining lines in the **/etc/anacrontab** file represent scheduled jobs and follow this format:

period in days	delay in minutes	job-identifier	command
-----------------------	-------------------------	-----------------------	----------------

- ▶ **period in days** — frequency of job execution in days

The property value can be defined as an integer or a macro (**@daily**, **@weekly**, **@monthly**), where **@daily** denotes the same value as integer **1**, **@weekly** the same as **7**, and **@monthly** specifies that the job is run once a month regardless of the length of the month.

- ▶ **delay in minutes** — number of minutes anacron waits before executing the job

The property value is defined as an integer. If the value is set to **0**, no delay applies.

- ▶ **job-identifier** — unique name referring to a particular job used in the log files

- ▶ **command** — command to be executed

The command can be either a command such as **ls /proc >> /tmp/proc** or a command which executes a custom script.

Any lines that begin with a hash sign (#) are comments and are not processed.

19.1.3.1. Examples of Anacron Jobs

The following example shows a simple **/etc/anacrontab** file:

```

SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=30
# the jobs will be started during the following hours only
START_HOURS_RANGE=16-20

#period in days    delay in minutes    job-identifier    command
1          20      dailyjob        nice run-parts /etc/cron.daily
7          25      weeklyjob       /etc/weeklyjob.bash
@monthly   45      monthlyjob     ls /proc >> /tmp/proc

```

All jobs defined in this **anacrontab** file are randomly delayed by 6-30 minutes and can be executed between 16:00 and 20:00.

The first defined job is triggered daily between 16:26 and 16:50 (RANDOM_DELAY is between 6 and 30 minutes; the delay in minutes property adds 20 minutes). The command specified for this job executes all present programs in the **/etc/cron.daily/** directory using the **run-parts** script (the **run-parts** scripts accepts a directory as a command-line argument and sequentially executes every program in the directory). See the **run-parts** for more information on the **run-parts** script.

The second job executes the **weeklyjob.bash** script in the **/etc/** directory once a week.

The third job runs a command, which writes the contents of **/proc** to the **/tmp/proc** file (**ls /proc >> /tmp/proc**) once a month.

19.1.4. Configuring Cron Jobs

The configuration file for cron jobs is **/etc/crontab**, which can be only modified by the **root** user. The file contains the following:

```

SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/
# For details see man 4 crontabs
# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .---- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .--- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * * username command to be executed

```

The first three lines contain the same variable definitions as an **anacrontab** file: **SHELL**, **PATH**, and **MAILTO**. For more information about these variables, see [Section 19.1.3, “Configuring Anacron Jobs”](#).

In addition, the file can define the **HOME** variable. The **HOME** variable defines the directory, which will be used as the home directory when executing commands or scripts run by the job.

The remaining lines in the **/etc/crontab** file represent scheduled jobs and have the following format:

minute	hour	day	month	day of week	username	command
--------	------	-----	-------	-------------	----------	---------

The following define the time when the job is to be run:

- ▶ **minute** — any integer from 0 to 59
- ▶ **hour** — any integer from 0 to 23
- ▶ **day** — any integer from 1 to 31 (must be a valid day if a month is specified)
- ▶ **month** — any integer from 1 to 12 (or the short name of the month such as jan or feb)
- ▶ **day of week** — any integer from 0 to 7, where 0 or 7 represents Sunday (or the short name of the week such as sun or mon)

The following define other job properties:

- ▶ **username** — specifies the user under which the jobs are run.
- ▶ **command** — the command to be executed.

The command can be either a command such as `ls /proc /tmp/proc` or a command which executes a custom script.

For any of the above values, an asterisk (*) can be used to specify all valid values. If you, for example, define the month value as an asterisk, the job will be executed every month within the constraints of the other values.

A hyphen (-) between integers specifies a range of integers. For example, **1-4** means the integers 1, 2, 3, and 4.

A list of values separated by commas (,) specifies a list. For example, **3,4,6,8** indicates exactly these four integers.

The forward slash (/) can be used to specify step values. The value of an integer will be skipped within a range following the range with **/integer**. For example, the minute value defined as **0-59/2** denotes every other minute in the minute field. Step values can also be used with an asterisk. For instance, if the month value is defined as *** /3**, the task will run every third month.

Any lines that begin with a hash sign (#) are comments and are not processed.

Users other than **root** can configure cron tasks with the **cron** utility. The user-defined crontabs are stored in the `/var/spool/cron/` directory and executed as if run by the users that created them.

To create a crontab as a specific user, login as that user and type the command **cron -e** to edit the user's crontab with the editor specified in the **VISUAL** or **EDITOR** environment variable. The file uses the same format as `/etc/crontab`. When the changes to the crontab are saved, the crontab is stored according to the user name and written to the file `/var/spool/cron/username`. To list the contents of the current user's crontab file, use the **cron -l** command.

The `/etc/cron.d/` directory contains files that have the same syntax as the `/etc/crontab` file. Only **root** is allowed to create and modify files in this directory.



Do not restart the daemon to apply the changes

The cron daemon checks the `/etc/anacrontab` file, the `/etc/crontab` file, the `/etc/cron.d/` directory, and the `/var/spool/cron/` directory every minute for changes and the detected changes are loaded into memory. It is therefore not necessary to restart the daemon after an anacrontab or a crontab file have been changed.

19.1.5. Controlling Access to Cron

To restrict the access to Cron, you can use the `/etc/cron.allow` and `/etc/cron.deny` files. These access control files use the same format with one user name on each line. Mind that no whitespace characters are permitted in either file.

If the `cron.allow` file exists, only users listed in the file are allowed to use cron, and the `cron.deny` file is ignored.

If the `cron.allow` file does not exist, users listed in the `cron.deny` file are not allowed to use Cron.

The Cron daemon (`crond`) does not have to be restarted if the access control files are modified. The access control files are checked each time a user tries to add or delete a cron job.

The `root` user can always use cron, regardless of the user names listed in the access control files.

You can control the access also through Pluggable Authentication Modules (PAM). The settings are stored in the `/etc/security/access.conf` file. For example, after adding the following line to the file, no other user but the `root` user can create crontabs:

```
- :ALL EXCEPT root :cron
```

The forbidden jobs are logged in an appropriate log file or, when using `crontab -e`, returned to the standard output. For more information, see `access.conf.5` (that is, `man 5 access.conf`).

19.1.6. Black and White Listing of Cron Jobs

Black and white listing of jobs is used to define parts of a job that do not need to be executed. This is useful when calling the `run-parts` script on a Cron directory, such as `/etc/cron.daily/`: if the user adds programs located in the directory to the job black list, the `run-parts` script will not execute these programs.

To define a black list, create a `jobs.deny` file in the directory that `run-parts` scripts will be executing from. For example, if you need to omit a particular program from `/etc/cron.daily/`, create the `/etc/cron.daily/jobs.deny` file. In this file, specify the names of the programs to be omitted from execution (only programs located in the same directory can be enlisted). If a job runs a command which runs the programs from the `/etc/cron.daily/` directory, such as `run-parts /etc/cron.daily`, the programs defined in the `jobs.deny` file will not be executed.

To define a white list, create a `jobs.allow` file.

The principles of `jobs.deny` and `jobs.allow` are the same as those of `cron.deny` and `cron.allow` described in section [Section 19.1.5, “Controlling Access to Cron”](#).

19.2. At and Batch

While Cron is used to schedule recurring tasks, the **At** utility is used to schedule a one-time task at a specific time and the **Batch** utility is used to schedule a one-time task to be executed when the system load average drops below 0.8.

19.2.1. Installing At and Batch

To determine if the *at* package is already installed on your system, issue the following command:

```
rpm -q at
```

The command returns the full name of the *at* package if already installed or notifies you that the package is not available.

To install the packages, use the **yum** command in the following form as **root**:

```
yum install package
```

For example, to install both At and Batch, type the following at a shell prompt:

```
~]# yum install at
```

For more information on how to install new packages in Red Hat Enterprise Linux, see [Section 5.2.4, “Installing Packages”](#).

19.2.2. Running the At Service

The At and Batch jobs are both picked by the **atd** service. This section provides information on how to start, stop, and restart the **atd** service, and shows how to configure it to start automatically at boot time. For more information on how to manage system services in Red Hat Enterprise Linux 7 in general, see [Chapter 6, Managing Services with systemd](#).

19.2.2.1. Starting and Stopping the At Service

To determine if the service is running, use the following command:

```
systemctl status atd.service
```

To run the **atd** service in the current session, type the following at a shell prompt as **root**:

```
systemctl start atd.service
```

To configure the service to start automatically at boot time, use the following command as **root**:

```
systemctl enable atd.service
```

Note

It is recommended that you configure your system to start the **atd** service automatically at boot time.

19.2.2.2. Stopping the At Service

To stop the **atd** service, type the following at a shell prompt as **root**:

```
systemctl stop atd.service
```

To prevent the service from starting automatically at boot time, use the following command as **root**:

```
systemctl disable atd.service
```

19.2.2.3. Restarting the At Service

To restart the **atd** service, type the following at a shell prompt as **root**:

```
systemctl restart atd.service
```

This command stops the service and starts it again in quick succession.

19.2.3. Configuring an At Job

To schedule a one-time job for a specific time with the **At** utility, do the following:

1. On the command line, type the command **at TIME**, where **TIME** is the time when the command is to be executed.

The **TIME** argument can be defined in any of the following formats:

- » **HH:MM** specifies the exact hour and minute; For example, **04:00** specifies 4:00 a.m.
- » **midnight** specifies 12:00 a.m.
- » **noon** specifies 12:00 p.m.
- » **teatime** specifies 4:00 p.m.
- » **MONTHDAYYEAR** format; For example, **January 15 2012** specifies the 15th day of January in the year 2012. The year value is optional.
- » **MMDDYY, MM/DD/YY, or MM.DD.YY** formats; For example, **011512** for the 15th day of January in the year 2012.
- » **now + TIME** where **TIME** is defined as an integer and the value type: minutes, hours, days, or weeks. For example, **now + 5 days** specifies that the command will be executed at the same time five days from now.

The time must be specified first, followed by the optional date. For more information about the time format, see the **/usr/share/doc/at-<version>/timespec** text file.

If the specified time has past, the job is executed at the time the next day.

2. In the displayed **at>** prompt, define the job commands:

- Type the command the job should execute and press **Enter**. Optionally, repeat the step to provide multiple commands.
- Enter a shell script at the prompt and press **Enter** after each line in the script.

The job will use the shell set in the user's **SHELL** environment, the user's login shell, or **/bin/sh** (whichever is found first).

3. Once finished, press **Ctrl+D** on an empty line to exit the prompt.

If the set of commands or the script tries to display information to standard output, the output is emailed to the user.

To view the list of pending jobs, use the **atq** command. See [Section 19.2.5, “Viewing Pending Jobs”](#) for more information.

You can also restrict the usage of the **at** command. For more information, see [Section 19.2.7, “Controlling Access to At and Batch”](#) for details.

19.2.4. Configuring a Batch Job

The **Batch** application executes the defined one-time tasks when the system load average decreases below 0.8.

To define a Batch job, do the following:

1. On the command line, type the command **batch**.
2. In the displayed **at>** prompt, define the job commands:
 - A. Type the command the job should execute and press **Enter**. Optionally, repeat the step to provide multiple commands.
 - B. Enter a shell script at the prompt and press **Enter** after each line in the script.

If a script is entered, the job uses the shell set in the user's **SHELL** environment, the user's login shell, or **/bin/sh** (whichever is found first).

3. Once finished, press **Ctrl+D** on an empty line to exit the prompt.

If the set of commands or the script tries to display information to standard output, the output is emailed to the user.

To view the list of pending jobs, use the **atq** command. See [Section 19.2.5, “Viewing Pending Jobs”](#) for more information.

You can also restrict the usage of the **batch** command. For more information, see [Section 19.2.7, “Controlling Access to At and Batch”](#) for details.

19.2.5. Viewing Pending Jobs

To view the pending **At** and **Batch** jobs, run the **atq** command. The **atq** command displays a list of pending jobs, with each job on a separate line. Each line follows the job number, date, hour, job class, and user name format. Users can only view their own jobs. If the **root** user executes the **atq** command, all jobs for all users are displayed.

19.2.6. Additional Command Line Options

Additional command line options for **at** and **batch** include the following:

Table 19.1. at and batch Command Line Options

Option	Description
-f	Read the commands or shell script from a file instead of specifying them at the prompt.
-m	Send email to the user when the job has been completed.
-v	Display the time that the job is executed.

19.2.7. Controlling Access to At and Batch

You can restrict the access to the **at** and **batch** commands using the **/etc/at.allow** and **/etc/at.deny** files. These access control files use the same format defining one user name on each line. Mind that no whitespace are permitted in either file.

If the file **at.allow** exists, only users listed in the file are allowed to use **at** or **batch**, and the **at.deny** file is ignored.

If **at.allow** does not exist, users listed in **at.deny** are not allowed to use **at** or **batch**.

The **at** daemon (**atd**) does not have to be restarted if the access control files are modified. The access control files are read each time a user tries to execute the **at** or **batch** commands.

The **root** user can always execute **at** and **batch** commands, regardless of the content of the access control files.

19.3. Additional Resources

To learn more about configuring automated tasks, see the following installed documentation:

- ▶ **cron(8)** man page contains an overview of cron.
- ▶ **crontab** man pages in sections 1 and 5:
 - The manual page in section 1 contains an overview of the **crontab** file.
 - The man page in section 5 contains the format for the file and some example entries.
- ▶ **anacron(8)** manual page contains an overview of anacron.
- ▶ **anacrontab(5)** manual page contains an overview of the **anacrontab** file.
- ▶ **run-parts(4)** manual page contains an overview of the **run-parts** script.
- ▶ **/usr/share/doc/at-<version>/timespec** contains detailed information about the time values that can be used in cron job definitions.
- ▶ **at** manual page contains descriptions of **at** and **batch** and their command line options.

Chapter 20. Automatic Bug Reporting Tool (ABRT)

20.1. Introduction to ABRT

The **Automatic Bug Reporting Tool**, commonly abbreviated as **ABRT**, is a set of tools that is designed to help users detect and report application crashes. Its main purpose is to ease the process of reporting issues and finding solutions. In this context, the solution can be a Bugzilla ticket, a knowledge-base article, or a suggestion to update a package to a version containing a fix.

ABRT consists of the **abrtd** daemon and a number of system services and utilities for processing, analyzing, and reporting detected problems. The daemon runs silently in the background most of the time and springs into action when an application crashes or a kernel oops is detected. The daemon then collects the relevant problem data, such as a core file if there is one, the crashing application's command line parameters, and other data of forensic utility.

ABRT currently supports the detection of crashes in applications written in the C/C++, Java, Python, and Ruby programming languages, as well as X.Org crashes, kernel oopses, and kernel panics. See [Section 20.4, “Detecting Software Problems”](#) for more detailed information on the types of crashes supported, and the way the various types of crashes are detected.

The identified problems can be reported to a remote issue tracker, and the reporting can be configured to happen automatically whenever an issue is detected. Problem data can also be stored locally or on a dedicated system and reviewed, reported, and deleted manually by the user. The reporting tools can send problem data to a Bugzilla database or the Red Hat Technical Support (RHT Support) site. The tools can also upload it using **FTP** or **SCP**, send it as an email, or write it to a file.

The **ABRT** component that handles already-existing problem data (as opposed to, for example, the creation of new problem data) is a part of a separate project, **libreport**. The **libreport** library provides a generic mechanism for analyzing and reporting problems, and it is used by applications other than **ABRT** as well. However, **ABRT** and **libreport** operation and configuration are closely integrated. They are, therefore, discussed as one in this document.

20.2. Installing ABRT and Starting its Services

In order to use **ABRT**, ensure that the *abrt-desktop* or the *abrt-cli* package is installed on your system. The *abrt-desktop* package provides a graphical user interface for **ABRT**, and the *abrt-cli* package contains a tool for using **ABRT** on the command line. You can also install both. The general workflow with both the **ABRT** GUI and the command line tool is procedurally similar and follows the same pattern.



Warning

Please note that installing the **ABRT** packages overwrites the **/proc/sys/kernel/core_pattern** file, which can contain a template used to name core-dump files. The content of this file will be overwritten to:

```
|/usr/libexec/abrt-hook-ccpp %s %c %p %u %g %t e
```

See [Section 5.2.4, “Installing Packages”](#) for general information on how to install packages using the **Yum** package manager.

20.2.1. Installing the ABRT GUI

The **ABRT** graphical user interface provides an easy-to-use front end for working in a desktop environment. You can install the required package by running the following command as the **root** user:

```
~]# yum install abrt-desktop
```

Upon installation, the **ABRT** notification applet is configured to start automatically when your graphical desktop session starts. You can verify that the **ABRT** applet is running by issuing the following command in a terminal:

```
~]$ ps -el | grep abrt-applet
0 S 500 2036 1824 0 80 0 - 61604 poll_s ? 00:00:00 abrt-applet
```

If the applet is not running, you can start it manually in your current desktop session by running the **abrt-applet** program:

```
~]$ abrt-applet &
[1] 2261
```

20.2.2. Installing ABRT for the Command Line

The *command line interface* lends itself to use on headless machines, remote systems connected over a network, or in scripts. You can install the required package by running the following command as the **root** user:

```
~]# yum install abrt-cli
```

20.2.3. Installing Supplementary ABRT Tools

To receive email notifications about crashes caught by **ABRT**, you need to have the *libreport-plugin-mailx* package installed. You can install it by executing the following command as **root**:

```
~]# yum install libreport-plugin-mailx
```

By default, it sends notifications to the **root** user at the local machine. The email destination can be configured in the */etc/libreport/plugins/mailx.conf* file.

To have notifications displayed in your console at login time, install the *abrt-console-notification* package as well.

Support for the various types of crashes that can be detected, analyzed, and reported by **ABRT** is contained in independent packages. Support for the most common types of crashes, such as crashes of C/C++ applications, is installed automatically when you install the basic **ABRT** system. See [Section 20.2, “Installing ABRT and Starting its Services”](#) for information on what packages need to be installed to obtain basic **ABRT** functionality.

However, packages that contain support for certain types of crashes need to be installed separately. For example, to install support for detecting exceptions in applications written using the Java language, run the following command as **root**:

```
~]# yum install abrt-java-connector
```

See [Section 20.4, “Detecting Software Problems”](#) for a list of languages and software projects with which **ABRT** can work. The section also includes a list of all corresponding packages that need to be installed for the detection of the various types of crashes to work.

20.2.4. Starting the ABRT Services

The **abrtd** daemon is configured to start at boot time. You can use the following command to verify its current status:

```
~]$ systemctl is-active abrtd.service
active
```

If the **systemctl** command returns **inactive** or **unknown**, the daemon is not running. It can be started for the current session by entering this command as the **root** user:

```
~]# systemctl start abrtd.service
```

Similarly, you can follow the same steps to check the status of and start the services that handle the various types of crashes. For example, make sure the **abrt-ccpp** service is running if you want **ABRT** to detect C/C++ crashes. See [Section 20.4, “Detecting Software Problems”](#) for a list of all available **ABRT** detection services and their respective packages.

With the exception of the **abrt-vmcore** and **abrt-pstoreoops** services, which are only started when a kernel panic or oops actually occurs, all **ABRT** services are automatically enabled and started at boot time when their respective packages are installed. You can disable or enable any **ABRT** service by using the **systemctl** utility as described in [Chapter 6, Managing Services with systemd](#).

20.2.5. Testing ABRT Crash Detection

To test that **ABRT** works properly, use the **kill** command to send the **SEGV** signal to terminate a process. For example, start a **sleep** process and terminate it with the **kill** command in the following way:

```
~]$ sleep 100 &
[1] 2823
~]$ kill -s SEGV 2823
```

ABRT detects a crash shortly after executing the **kill** command, and, provided a graphical session is running, the user is notified of the detected problem by the GUI notification applet. In the command line environment, you can check that the crash was detected by running the **abrt-cli list** command or by examining the crash dump created in the **/var/tmp/abrt/** directory. See [Section 20.5, “Handling Detected Problems”](#) for more information on how to work with detected crashes.

20.3. Configuring ABRT

A problem life cycle is driven by events in **ABRT**. For example:

- ▶ Event #1 — a problem-data directory is created.
- ▶ Event #2 — problem data is analyzed.
- ▶ Event #3 — the problem is reported to Bugzilla.

Whenever a problem is detected, **ABRT** compares it with all existing problem data and determines whether that same problem has already been recorded. If it has, the existing problem data is updated, and the most recent (duplicate) problem is not recorded again. If the problem is not recognized by **ABRT**, a **problem-data directory** is created. A problem-data directory typically consists of files such as: **analyzer**, **architecture**, **coredump**, **cmdline**, **executable**, **kernel**, **os_release**, **reason**, **time**, and **uid**.

Other files, such as **backtrace**, can be created during the analysis of the problem, depending on which analyzer method is used and its configuration settings. Each of these files holds specific information about the system and the problem itself. For example, the **kernel** file records the version of a crashed kernel.

After the problem-data directory is created and problem data gathered, you can process the problem using either the **ABRT** GUI, or the **abrt-cli** utility for the command line. See [Section 20.5, “Handling Detected Problems”](#) for more information about the **ABRT** tools provided for working with recorded problems.

20.3.1. Configuring Events

ABRT events use *plugins* to carry out the actual reporting operations. Plugins are compact utilities that the events call to process the content of problem-data directories. Using plugins, **ABRT** is capable of reporting problems to various destinations, and almost every reporting destination requires some configuration. For instance, Bugzilla requires a username, password, and a URL pointing to an instance of the Bugzilla service.

Some configuration details can have default values (such as a Bugzilla URL), but others cannot have sensible defaults (for example, a username). **ABRT** looks for these settings in configuration files, such as **report_Bugzilla.conf**, in the **/etc/libreport/events/** or **\$HOME/.cache/abrt/events/** directories for system-wide or user-specific settings respectively. The configuration files contain pairs of directives and values.

These files are the bare minimum necessary for running events and processing the problem-data directories. The **abrt-gui** and **abrt-cli** tools read the configuration data from these files and pass it to the events they run.

Additional information about events (such as their description, names, types of parameters that can be passed to them as environment variables, and other properties) is stored in **event_name.xml** files in the **/usr/share/libreport/events/** directory. These files are used by both **abrt-gui** and **abrt-cli** to make the user interface more friendly. Do not edit these files unless you want to modify the standard installation. If you intend to do that, copy the file to be modified to the **/etc/libreport/events/** directory and modify the new file. These files can contain the following information:

- ▶ a user-friendly event name and description (Bugzilla, Report to Bugzilla bug tracker),
- ▶ a list of items in a problem-data directory that are required for the event to succeed,
- ▶ a default and mandatory selection of items to send or not send,
- ▶ whether the GUI should prompt for data review,
- ▶ what configuration options exist, their types (string, Boolean, etc.), default value, prompt string, etc.; this lets the GUI build appropriate configuration dialogs.

For example, the **report_Logger** event accepts an output filename as a parameter. Using the respective **event_name.xml** file, the **ABRT** GUI determines which parameters can be specified for a selected event and allows the user to set the values for these parameters. The values are saved by the **ABRT** GUI and reused on subsequent invocations of these events. Note that the **ABRT** GUI saves configuration options using the **GNOME Keyring** tool and by passing them to events, it overrides data from text configuration files.

To open the graphical **Configuration** window, choose **Automatic Bug Reporting Tool → Preferences** from within a running instance of the **abrt-gui** application. This window shows a list of events that can be selected during the reporting process when using the GUI. When you select one of the configurable events, you can click the **Configure** button and modify the settings for that event.



Important

All files in the `/etc/libreport/` directory hierarchy are world-readable and are meant to be used as global settings. Thus, it is not advisable to store usernames, passwords, or any other sensitive data in them. The per-user settings (set in the GUI application and readable by the owner of `$HOME` only) are safely stored in **GNOOME Keyring**, or they can be stored in a text configuration file in `$HOME/.abrt/` for use with **abrt-cli**.

The following table shows a selection of the default analyzing, collecting, and reporting events provided by the standard installation of **ABRT**. The table lists each event's name, identifier, configuration file from the `/etc/libreport/events.d/` directory, and a brief description. Note that while the configuration files use the event identifiers, the **ABRT** GUI refers to the individual events using their names. Note also that not all of the events can be set up using the GUI. For information on how to define a custom event, see [Section 20.3.2, “Creating Custom Events”](#).

Table 20.1. Standard ABRT Events

Name	Identifier and Configuration File	Description
uReport	report_uReport	Uploads a µReport to the FAF server.
Mailx	report_Mailx mailx_event.conf	Sends the problem report via the Mailx utility to a specified email address.
Bugzilla	report_Bugzilla bugzilla_event.conf	Reports the problem to the specifid installation of the Bugzilla bug tracker.
Red Hat Customer Support	report_RHTSupport rhtsupport_event.conf	Reports the problem to the Red Hat Technical Support system.
Emergency analysis	report_EmergencyAnalysis emergencyanalysis_event.conf	Uploads a tarball to the FAF server for further analysis. Used in case of a reporting failure when standard reporting methods fail.
Analyze C/C++ Crash	analyze_CCpp ccpp_event.conf	Sends the core dump to a remote retrace server for analysis or performs a local analysis if the remote one fails.
Report uploader	report_Uploader uploader_event.conf	Uploads a tarball (<code>.tar.gz</code>) archive with problem data to the chosen destination using the FTP or the SCP protocol.
Analyze VM core	analyze_VMcore vmcore_event.conf	Runs the GDB (the GNU debugger) on the problem data of a kernel oops and generates a backtrace of the kernel.

Name	Identifier and Configuration File	Description
Local GNU Debugger	analyze_LocalGDB ccpp_event.conf	Runs GDB (the GNU debugger) on the problem data of an application and generates a backtrace of the program.
Collect .xsession-errors	analyze_xsession_errors ccpp_event.conf	Saves relevant lines from the <code>~/.xsession-errors</code> file to the problem report.
Logger	report_Logger print_event.conf	Creates a problem report and saves it to a specified local file.
Kerneloops.org	report_Kerneloops koops_event.conf	Sends a kernel problem to the oops tracker at kerneloops.org .

20.3.2. Creating Custom Events

Each event is defined by one rule structure in a respective configuration file. The configuration files are typically stored in the `/etc/libreport/events.d/` directory. These configuration files are loaded by the main configuration file, `/etc/libreport/report_event.conf`.

If you would like to modify this file, please note that it respects shell metacharacters (*, \$, ?, etc.) and interprets relative paths relatively to its location.

Each *rule* starts with a line with a non-space leading character, and all subsequent lines starting with the **space** character or the **tab** character are considered a part of this rule. Each *rule* consists of two parts, a *condition* part and a *program* part. The condition part contains conditions in one of the following forms:

- ▶ `VAR=VAL`
- ▶ `VAR!=VAL`
- ▶ `VAL~=REGEX`

...where:

- ▶ `VAR` is either the **EVENT** key word or a name of a problem-data directory element (such as **executable**, **package**, **hostname**, etc.),
- ▶ `VAL` is either a name of an event or a problem-data element, and
- ▶ `REGEX` is a regular expression.

The program part consists of program names and shell-interpretable code. If all conditions in the condition part are valid, the program part is run in the shell. The following is an event example:

```
EVENT=post-create date > /tmp/dt
echo $HOSTNAME `uname -r`
```

This event would overwrite the contents of the `/tmp/dt` file with the current date and time and print the hostname of the machine and its kernel version on the standard output.

Here is an example of a more complex event, which is actually one of the predefined events. It saves relevant lines from the `~/.xsession-errors` file to the problem report of any problem for which the **abrt-ccpp** service has been used, provided the crashed application had any X11 libraries loaded at the time of the crash:

```
EVENT=analyze_xsession_errors analyzer=CCpp dso_list=~.*libX11.*
    test -f ~/.xsession-errors || { echo "No ~/.xsession-errors"; exit 1; }
    test -r ~/.xsession-errors || { echo "Can't read ~/.xsession-errors"; exit
1; }
    executable=`cat executable` &&
    base_executable=${executable##*/} &&
    grep -F -e "$base_executable" ~/.xsession-errors | tail -999
>xsession_errors &&
    echo "Element 'xsession_errors' saved"
```

The set of possible events is not definitive. System administrators can add events according to their need. Currently, the following event names are provided with the standard **ABRT** and **libreport** installations:

post-create

This event is run by **abrtd** to process newly created problem-data directories. When the **post-create** event is run, **abrtd** checks whether the **id** identifier of the new problem data matches the **id** of any already existing problem directories. If such a problem directory exists, the new problem data is deleted.

notify, notify-dup

The **notify** event is run following the completion of **post-create**. When the event is run, the user can be sure that the problem deserves their attention. The **notify-dup** is similar, except it is used for duplicate occurrences of the same problem.

analyze_name_suffix

...where *name_suffix* is the adjustable part of the event name. This event is used to process collected data. For example, the **analyze_LocalGDB** event uses the GNU Debugger (**GDB**) utility to process the core dump of an application and produce a backtrace of the crash.

collect_name_suffix

...where *name_suffix* is the adjustable part of the event name. This event is used to collect additional information on problems.

report_name_suffix

...where *name_suffix* is the adjustable part of the event name. This event is used to report a problem.

20.3.3. Setting Up Automatic Reporting

ABRT can be configured to send initial anonymous reports, or *μReports*, of any detected issues or crashes automatically without any user interaction. When automatic reporting is turned on, the so called *μReport*, which is normally sent at the beginning of the crash-reporting process, is sent immediately after a crash is detected. This allows for an efficient way of preventing duplicate support cases based on identical crashes. To enable the autoreporting feature, issue the following command as **root**:

```
~]# abrt-auto-reporting enabled
```

The above command sets the **AutoreportingEnabled** directive in the `/etc/abrt/abrt.conf` configuration file to **yes**. This system-wide setting applies to all users of the system. Note that by enabling this option, automatic reporting will also be enabled in the graphical desktop environment. To only enable autoreporting in the **ABRT** GUI, switch the **Automatically send uReport** option to **YES** in the **Problem Reporting Configuration** window. To open this window, choose **Automatic Bug Reporting Tool** → **ABRT Configuration** from within a running instance of the **abrt-gui** application. To launch the application, go to **Applications** → **Sundry** → **Automatic Bug Reporting Tool**.

Upon detection of a crash, by default, **ABRT** submits a **μReport** with basic information about the problem to Red Hat's **ABRT** server. The server determines whether the problem is known and either provides a short description of the problem along with a URL of the reported case if known, or invites the user to report it if not known.



Note

A **μReport** (microreport) is a JSON object representing a problem, such as a binary crash or a kernel oops. These reports are designed to be small, machine readable, and completely anonymous, which is why they can be used for automated reporting. The **μReports** make it possible to keep track of bug occurrences, but they usually do not provide enough information for engineers to fix the bug. A full bug report is needed for a support case to be opened.

To change the default behavior of the autoreporting facility from sending a **μReport** to something else, you need to modify the value of the **AutoreportingEvent** directive in the `/etc/abrt/abrt.conf` configuration file to point to a different **ABRT** event. See [Table 20.1, “Standard ABRT Events”](#) for an overview of the standard events.

20.4. Detecting Software Problems

ABRT is capable of detecting, analyzing, and processing crashes in applications written in a variety of different programming languages. Many of the packages that contain the support for detecting the various types of crashes are installed automatically when either one of the main **ABRT** packages (`abrt-desktop`, `abrt-cli`) is installed. See [Section 20.2, “Installing ABRT and Starting its Services”](#) for instructions on how to install **ABRT**. See the table below for a list of the supported types of crashes and the respective packages.

Table 20.2. Supported Programming Languages and Software Projects

Langauge/Project	Package
C/C++	<code>abrt-addon-ccpp</code>
Python	<code>abrt-addon-python</code>
Ruby	<code>rubygem-abrt</code>
Java	<code>abrt-java-connector</code>
X.Org	<code>abrt-addon-xorg</code>
Linux (kernel oops)	<code>abrt-addon-kerneloops</code>
Linux (kernel panic)	<code>abrt-addon-vmcore</code>
Linux (persistent storage)	<code>abrt-addon-pstoreoops</code>

20.4.1. Detecting C/C++ Crashes

The **abrt-ccpp** service installs its own core-dump handler, which, when started, overrides the default value of the kernel's **core_pattern** variable, so that C/C++ crashes are handled by **abrt**. If you stop the **abrt-ccpp** service, the previously specified value of **core_pattern** is reinstated.

By default, the **/proc/sys/kernel/core_pattern** file contains the string **core**, which means that the kernel produces files named **core.*** in the current directory of the crashed process. The **abrt-ccpp** service overwrites the **core_pattern** file to contain the following command:

```
| /usr/libexec/abrt-hook-ccpp %s %c %p %u %g %t e
```

This command instructs the kernel to pipe the core dump to the **abrt-hook-ccpp** program, which stores it in **ABRT**'s dump location and notifies the **abrt** daemon of the new crash. It also stores the following files from the **/proc/PID/** directory (where *PID* is the ID of the crashed process) for debugging purposes: **maps**, **limits**, **cgroup**, **status**. See **proc(5)** for a description of the format and the meaning of these files.

20.4.2. Detecting Python Exceptions

The **abrt-addon-python** package installs a custom exception handler for Python applications. The Python interpreter then automatically imports the **abrt.pth** file installed in **/usr/lib64/python2.7/site-packages/**, which in turn imports **abrt_exception_handler.py**. This overrides Python's default **sys.excepthook** with a custom handler, which forwards unhandled exceptions to **abrt** via its Socket API.

To disable the automatic import of site-specific modules, and thus prevent the **ABRT** custom exception handler from being used when running a Python application, pass the **-S** option to the Python interpreter:

```
|~]$ python -S file.py
```

In the above command, replace *file.py* with the name of the Python script you want to execute without the use of site-specific modules.

20.4.3. Detecting Ruby Exceptions

The **rubygem-abrt** package registers a custom handler using the **at_exit** feature, which is executed when a program ends. This allows for checking for possible unhandled exceptions. Every time an unhandled exception is captured, the **ABRT** handler prepares a bugreport, which can be submitted to Red Hat Bugzilla using standard **ABRT** tools.

20.4.4. Detecting Java Exceptions

The **ABRT** Java Connector is a JVM agent that reports uncaught Java exceptions to **abrt**. The agent registers several JVMTI event callbacks and has to be loaded into the JVM using the **-agentlib** command line parameter. Note that the processing of the registered callbacks negatively impacts the performance of the application. Use the following command to have **ABRT** catch exceptions from a Java class:

```
|~]$ java -agentlib:abrt-java-connector[=abrt=on] $MyClass -platform.jvmtiSupported true
```

In the above command, replace *\$MyClass* with the name of the Java class you want to test. By passing the **abrt=on** option to the connector, you ensure that the exceptions are handled by **abrt**. In case you wish to have the connector output the exceptions to standard output, omit this option.

20.4.5. Detecting X.Org Crashes

The **abrt-xorg** service collects and processes information about crashes of the **X.Org server** from the **/var/log/Xorg.0.log** file. Note that no report is generated if a blacklisted **X.org** module is loaded. Instead, a **not-reportable** file is created in the problem-data directory with an appropriate explanation. You can find the list of offending modules in the **/etc/abrt/plugins/xorg.conf** file. Only proprietary graphics-driver modules are blacklisted by default.

20.4.6. Detecting Kernel Ooops and Panics

By checking the output of kernel logs, **ABRT** is able to catch and process the so-called kernel oopses — non-fatal deviations from the correct behavior of the Linux kernel. This functionality is provided by the **abrt-oops** service.

ABRT can also detect and process kernel panics, that is fatal, non-recoverable errors that require a reboot, using the **abrt-vmcore** service. The service only starts when a **vmcore** file (a kernel-core dump) appears in the **/var/crash/** directory. When a core-dump file is found, **abrt-vmcore** creates a new **problem-data directory** in the **/var/tmp/abrt/** directory and moves the core-dump file to the newly created problem-data directory. After the **/var/crash/** directory is searched, the service is stopped.

For **ABRT** to be able to detect a kernel panic, the **kdump** service must be enabled on the system. The amount of memory that is reserved for the kdump kernel has to be set correctly. You can set it using the **system-config-kdump** graphical tool or by specifying the **crashkernel** parameter in the list of kernel options in the **/etc/default/grub** configuration file. For details on how to enable and configure **kdump**, see the [Red Hat Enterprise Linux 7 Kernel Crash Dump Guide](#).

Using the **abrt-pstoreoops** service, **ABRT** is capable of collecting and processing innformation about kernel panics, which, on systems that support **pstore**, is stored in the automatically-mounted **/sys/fs/pstore/** directory. The service starts automatically when kernel crash-dump files appear in the **/sys/fs/pstore/** directory.

20.5. Handling Detected Problems

Problem data saved by **abrt** can be viewed, reported, and deleted using either the command line tool, **abrt-cli**, or the graphical tool, **abrt-gui**.



Note

Note that **ABRT** identifies duplicate problems by comparing new problems with all locally saved problems. For a repeating crash, **ABRT** requires you to act upon it only once. However, if you delete the crash dump of that problem, the next time this specific problem occurs, **ABRT** will treat it as a new crash: **ABRT** will alert you about it, prompt you to fill in a description, and report it. To avoid having **ABRT** notifying you about a recurring problem, do not delete its problem data.

20.5.1. Using the Command Line Tool

In the command line environment, the user is notified of new crashes on login, provided they have the **abrt-console-notification** package installed. The console notification looks like the following:

```
ABRT has detected 1 problem(s). For more info run: abrt-cli list --since 1398783164
```

To view detected problems, enter the **abrt-cli list** command:

```
~]$ abrt-cli list
id 6734c6f1a1ed169500a7bfc8bd62aabaf039f9aa
Directory:      /var/tmp/abrt/ccpp-2014-04-21-09:47:51-3430
count:          1
executable:    /usr/bin/sleep
package:        coreutils-8.22-11.el7
time:           Mon 21 Apr 2014 09:47:51 AM EDT
uid:            1000
Run 'abrt-cli report /var/tmp/abrt/ccpp-2014-04-21-09:47:51-3430' for creating a
case in Red Hat Customer Portal
```

Each crash listed in the output of the **abrt-cli list** command has a unique identifier and a directory that can be used for further manipulation using **abrt-cli**.

To view information about just one particular problem, use the **abrt-cli info** command:

```
abrt-cli info [-d] directory_or_id
```

To increase the amount of information displayed when using both the **list** and **info** sub-commands, pass them the **-d (--detailed)** option, which shows all stored information about the problems listed, including respective **backtrace** files if they have already been generated.

To analyze and report a certain problem, use the **abrt-cli report** command:

```
abrt-cli report directory_or_id
```

Upon invocation of the above command, you will be asked to provide your credentials for opening a support case with Red Hat Customer Support. Next, **abrt-cli** opens a text editor with the content of the report. You can see what is being reported, and you can fill in instructions on how to reproduce the crash and other comments. You should also check the backtrace because the backtrace might be sent to a public server and viewed by anyone, depending on the problem-reporter event settings.

Note

You can choose which text editor is used to check the reports. **abrt-cli** uses the editor defined in the **ABRT_EDITOR** environment variable. If the variable is not defined, it checks the **VISUAL** and **EDITOR** variables. If none of these variables is set, the **vi** editor is used. You can set the preferred editor in your **.bashrc** configuration file. For example, if you prefer **GNU Emacs**, add the following line to the file:

```
export VISUAL=emacs
```

When you are done with the report, save your changes and close the editor. If you have reported your problem to the Red Hat Customer Support database, a problem case is filed in the database. From now on, you will be informed about the problem-resolution progress via email you provided during the process of reporting. You can also oversee the problem case using the URL that is provided to you when the problem case is created or via emails received from Red Hat Support.

If you are certain that you do not want to report a particular problem, you can delete it. To delete a problem, so that **ABRT** does not keep information about it, use the command:

```
abrt-cli rm directory_or_id
```

To display help about a particular **abrt-cli** command, use the **--help** option:

```
abrt-cli command --help
```

20.5.2. Using the GUI

The **ABRT** daemon broadcasts a **D-Bus** message whenever a problem report is created. If the **ABRT** applet is running in a graphical desktop environment, it catches this message and displays a notification dialog on the desktop. You can open the **ABRT** GUI using this dialog by clicking on the **Report** button. You can also open the **ABRT** GUI by selecting the **Applications → Sundry → Automatic Bug Reporting Tool** menu item.

Alternatively, you can run the **ABRT** GUI from the command line as follows:

```
~]$ abrt-gui &
```

The **ABRT** GUI window displays a list of detected problems. Each problem entry consists of the name of the failing application, the reason why the application crashed, and the date of the last occurrence of the problem.

To access a detailed problem description, double-click on a problem-report line or click on the **Report** button while the respective problem line is selected. You can then follow the instructions to proceed with the process of describing the problem, determining how it should be analyzed, and where it should be reported. To discard a problem, click on the **Delete** button.

20.6. Additional Resources

For more information about **ABRT** and related topics, see the resources listed below.

Installed Documentation

- ▶ **abrtd(8)** — The manual page for the **abrtd** daemon provides information about options that can be used with the daemon.
- ▶ **abrt_event.conf(5)** — The manual page for the **abrt_event.conf** configuration file describes the format of its directives and rules and provides reference information about event meta-data configuration in XML files.

Online Documentation

- ▶ [Red Hat Enterprise Linux 7 Networking Guide](#) — The *Networking Guide* for Red Hat Enterprise Linux 7 documents relevant information regarding the configuration and administration of network interfaces and network services on this system.
- ▶ [Red Hat Enterprise Linux 7 Kernel Crash Dump Guide](#) — The *Kernel Crash Dump Guide* for Red Hat Enterprise Linux 7 documents how to configure, test, and use the **kdump** crash recovery service and provides a brief overview of how to analyze the resulting core dump using the **crash** debugging utility.

See Also

- ▶ [Chapter 18, Viewing and Managing Log Files](#) describes the configuration of the **rsyslog** daemon and explains how to locate, view, and monitor log files.

- ▶ [Chapter 5, Yum](#) describes how to use the **Yum** package manager to search, install, update, and uninstall packages on the command line.
- ▶ [Chapter 6, Managing Services with systemd](#) provides an introduction to **systemd** and documents how to use the **systemctl** command to manage system services, configure systemd targets, and execute power management commands.

Chapter 21. OProfile

OProfile is a low overhead, system-wide performance monitoring tool. It uses the performance monitoring hardware on the processor to retrieve information about the kernel and executables on the system, such as when memory is referenced, the number of L2 cache requests, and the number of hardware interrupts received. On a Red Hat Enterprise Linux system, the **oprofile** package must be installed to use this tool.

Many processors include dedicated performance monitoring hardware. This hardware makes it possible to detect when certain events happen (such as the requested data not being in cache). The hardware normally takes the form of one or more *counters* that are incremented each time an event takes place. When the counter value increments, an interrupt is generated, making it possible to control the amount of detail (and therefore, overhead) produced by performance monitoring.

OProfile uses this hardware (or a timer-based substitute in cases where performance monitoring hardware is not present) to collect *samples* of performance-related data each time a counter generates an interrupt. These samples are periodically written out to disk; later, the data contained in these samples can then be used to generate reports on system-level and application-level performance.

OProfile is a useful tool, but be aware of some limitations when using it:

- ▶ *Use of shared libraries* — Samples for code in shared libraries are not attributed to the particular application unless the **--separate=library** option is used.
- ▶ *Performance monitoring samples are inexact* — When a performance monitoring register triggers a sample, the interrupt handling is not precise like a divide by zero exception. Due to the out-of-order execution of instructions by the processor, the sample may be recorded on a nearby instruction.
- ▶ ***opreport*** *does not associate samples for inline functions properly* — **opreport** uses a simple address range mechanism to determine which function an address is in. Inline function samples are not attributed to the inline function but rather to the function the inline function was inserted into.
- ▶ *OProfile accumulates data from multiple runs* — OProfile is a system-wide profiler and expects processes to start up and shut down multiple times. Thus, samples from multiple runs accumulate. Use the command **opcontrol --reset** to clear out the samples from previous runs.
- ▶ *Hardware performance counters do not work on guest virtual machines* — Because the hardware performance counters are not available on virtual systems, you need to use the **timer** mode. Run the command **opcontrol --deinit**, and then execute **modprobe oprofile timer=1** to enable the **timer** mode.
- ▶ *Non-CPU-limited performance problems* — OProfile is oriented to finding problems with CPU-limited processes. OProfile does not identify processes that are asleep because they are waiting on locks or for some other event to occur (for example an I/O device to finish an operation).

21.1. Overview of Tools

[Table 21.1, “OProfile Commands”](#) provides a brief overview of the most often used tools provided with the **oprofile** package.

Table 21.1. OProfile Commands

Command	Description
ophelp	Displays available events for the system's processor along with a brief description of each.

Command	Description
opimport	Converts sample database files from a foreign binary format to the native format for the system. Only use this option when analyzing a sample database from a different architecture.
opannotate	Creates annotated source for an executable if the application was compiled with debugging symbols. See Section 21.6.4, “Using opannotate” for details.
opcontrol	Configures what data is collected. See Section 21.3, “Configuring OProfile Using Legacy Mode” for details.
operf	Recommended tool to be used in place of opcontrol for profiling. See Section 21.2, “Using operf” for details. For differences between operf and opcontrol see Section 21.1.1, “operf vs. opcontrol” .
opreport	Retrieves profile data. See Section 21.6.1, “Using opreport” for details.
oprofiled	Runs as a daemon to periodically write sample data to disk.

21.1.1. operf vs. opcontrol

There are two mutually exclusive methods for collecting profiling data with OProfile. Either you can use the newer and preferred **operf** or the **opcontrol** tool.

operf

This is the recommended mode for profiling. The **operf** tool uses the Linux Performance Events Subsystem, and therefore does not require the *oprofile* kernel driver. The **operf** tool allows you to target your profiling more precisely, as a single process or system-wide, and also allows OProfile to co-exist better with other tools using the performance monitoring hardware on your system. Unlike **opcontrol**, it can be used without the **root** privileges. However, **operf** is also capable of system-wide operations with use of the **--system-wide** option, where **root** authority is required.

With **operf**, there is no initial setup needed. You can invoke **operf** with command-line options to specify your profiling settings. After that, you can run the OProfile post-processing tools described in [Section 21.6, “Analyzing the Data”](#). See [Section 21.2, “Using operf”](#) for further information.

Legacy Mode

This mode consists of the **opcontrol** shell script, the **oprofiled** daemon, and several post-processing tools. The **opcontrol** command is used for configuring, starting, and stopping a profiling session. An OProfile kernel driver, usually built as a kernel module, is used for collecting samples, which are then recorded into sample files by **oprofiled**. You can use legacy mode only if you have **root** privileges. In certain cases, such as when you need to sample areas with disabled interrupt request (IRQ), this is a better alternative.

Before OProfile can be run in legacy mode, it must be configured as shown in [Section 21.3, “Configuring OProfile Using Legacy Mode”](#). These settings are then applied when starting OProfile ([Section 21.4, “Starting and Stopping OProfile Using Legacy Mode”](#)).

21.2. Using operf

As mentioned before, **operf** is the recommended profiling mode that does not require an initial setup before starting. All settings are specified as command-line options and there is no separate command to start the profiling process. To stop **operf**, press Ctrl-c. The typical **operf** command syntax looks as follows:

```
operf options range command args
```

Replace *options* with desired command-line options to specify your profiling settings. Full set of options is described in **operf** man pages. Replace *range* with one of the following:

--system-wide - this setting allows for global profiling, see [Using operf in System-wide Mode](#)

--pid=PID - this is to profile running application, where *PID* is process ID of the process you wish to profile.

With *command* and *args*, you can define a specific command or application to be profiled, and also the input arguments that this command or application requires. Either *command*, **--pid** or **--system-wide** is required, but these cannot be used simultaneously.

When you invoke **operf** on a command line without setting the *range* option, data will be collected for the children processes.



Using **operf** in System-wide Mode

To run **operf --system-wide**, you need **root** authority. When finished profiling, you can stop **operf** with:

```
Ctrl-C
```

If you run **operf --system-wide** as a background process (with &), stop it in a controlled manner in order to process the collected profile data. For this purpose, use:

```
kill -SIGINT operf-PID
```

When running **operf --system-wide**, it is recommended that your current working directory is **/root** or a subdirectory of **/root** so that sample data files are not stored in locations accessible by regular users.

21.2.1. Specifying the Kernel

To monitor the kernel, execute the following command:

```
operf --vmlinuz=vmlinuX_path
```

With this option, you can specify a path to a vmlinuX file that matches the running kernel. Kernel samples will be attributed to this binary, allowing post-processing tools to attribute samples to the appropriate kernel symbols. If this option is not specified, all kernel samples will be attributed to a pseudo binary named "no-vmlinuX".

21.2.2. Setting Events to Monitor

Most processors contain counters, which are used by OProfile to monitor specific events. As shown in [Table 21.3, “OProfile Processors and Counters”](#), the number of counters available depends on the processor.

The events for each counter can be configured via the command line or with a graphical interface. For more information on the graphical interface, see [Section 21.10, “Graphical Interface”](#). If the counter cannot be set to a specific event, an error message is displayed.

Older Processors and `operf`

Some older processor models are not supported by the underlying Linux Performance Events Subsystem kernel and therefore are not supported by `operf`. If you receive this message:

```
Your kernel's Performance Events Subsystem does not support your processor type
```

when attempting to use `operf`, try profiling with `opcontrol` to see if your processor type may be supported by OProfile's legacy mode.

Using `operf` on Virtual Systems

Since hardware performance counters are not available on guest virtual machines, you have to enable *timer* mode to use `operf` on virtual systems. To do so, type as **root**:

```
opcontrol --deinit
```

```
modprobe oprofile timer=1
```

To set the event for each configurable counter via the command line, use:

```
operf --events=event1,event2...
```

Here, pass a comma-separated list of event specifications for profiling. Each event specification is a colon-separated list of attributes in the following form:

```
event-name:sample-rate:unit-mask:kernel:user
```

[Table 21.2, “Event Specifications”](#) summarizes these options. The last three values are optional, if you omit them, they will be set to their default values. Note that certain events do require a unit mask.

Table 21.2. Event Specifications

Specification	Description
<code>event-name</code>	The exact symbolic event name taken from <code>ophelp</code>

Specification	Description
<i>sample-rate</i>	The number of events to wait before sampling again. The smaller the count, the more frequent the samples. For events that do not happen frequently, a lower count may be needed to capture a statistically significant number of event instances. On the other hand, sampling too frequently can overload the system. By default, OProfile uses a time-based event set, which creates a sample every 100,000 clock cycles per processor.
<i>unit-mask</i>	Unit masks, which further define the event, are listed in ophelp . You can insert either a hexadecimal value, beginning with "0x", or a string that matches the first word of the unit mask description in ophelp . The second option is valid only for unit masks having "extra:" parameters, as shown by the output of ophelp . This type of unit mask cannot be defined with a hexadecimal value.
<i>kernel</i>	Specifies whether to profile kernel code (insert 0 or 1 (default))
<i>user</i>	Specifies whether to profile user-space code (insert 0 or 1 (default))

The events available vary depending on the processor type. When no event specification is given, the default event for the running processor type will be used for profiling. See [Table 21.4, “Default Events”](#) for a list of these default events. To determine the events available for profiling, use the **ophelp** command.

ophelp

21.2.3. Categorization of Samples

The **--separate-thread** option categorizes samples by thread group ID (tgid) and thread ID (tid). This is useful for seeing per-thread samples in multi-threaded applications. When used in conjunction with the **-system-wide** option, **--separate-thread** is also useful for seeing per-process (i.e., per-thread group) samples for the case where multiple processes are executing the same program during a profiling run.

The **--separate-cpu** option categorizes samples by cpu.

21.3. Configuring OProfile Using Legacy Mode

Before OProfile can be run in legacy mode, it must be configured. At a minimum, selecting to monitor the kernel (or selecting not to monitor the kernel) is required. The following sections describe how to use the **opcontrol** utility to configure OProfile. As the **opcontrol** commands are executed, the setup options are saved to the **/root/.oprofile/daemonrc** file.

21.3.1. Specifying the Kernel

First, configure whether OProfile should monitor the kernel. This is the only configuration option that is required before starting OProfile. All others are optional.

To monitor the kernel, execute the following command as **root**:

```
opcontrol --setup --vmlinux=/usr/lib/debug/lib/modules/`uname -r`/vmlinux
```



Install the debuginfo package

The **debuginfo** package for the kernel must be installed (which contains the uncompressed kernel) in order to monitor the kernel.

To configure OProfile not to monitor the kernel, execute the following command as **root**:

```
opcontrol --setup --no-vmlinux
```

This command also loads the **oprofile** kernel module, if it is not already loaded, and creates the **/dev/oprofile/** directory, if it does not already exist. See [Section 21.7, “Understanding /dev/oprofile/”](#) for details about this directory.

Setting whether samples should be collected within the kernel only changes what data is collected, not how or where the collected data is stored. To generate different sample files for the kernel and application libraries, see [Section 21.3.3, “Separating Kernel and User-space Profiles”](#).

21.3.2. Setting Events to Monitor

Most processors contain *counters*, which are used by OProfile to monitor specific events. As shown in [Table 21.3, “OProfile Processors and Counters”](#), the number of counters available depends on the processor.

Table 21.3. OProfile Processors and Counters

Processor	cpu_type	Number of Counters
AMD64	x86-64/hammer	4
AMD Family 10h	x86-64/family10	4
AMD Family 11h	x86-64/family11	4
AMD Family 12h	x86-64/family12	4
AMD Family 14h	x86-64/family14	4
AMD Family 15h	x86-64/family15	6
IBM eServer System i and IBM eServer System p	timer	1
IBM POWER4	ppc64/power4	8
IBM POWER5	ppc64/power5	6
IBM PowerPC 970	ppc64/970	8
IBM PowerPC 970MP	ppc64/970MP	8
IBM POWER5+	ppc64/power5+	6
IBM POWER5++	ppc64/power5++	6
IBM POWER56	ppc64/power6	6
IBM POWER7	ppc64/power7	6
IBM S/390 and IBM System z	timer	1
Intel Core i7	i386/core_i7	4
Intel Nehalem microarchitecture	i386/nehalem	4

Processor	cpu_type	Number of Counters
Intel Westmere microarchitecture	i386/westmere	4
Intel Haswell microarchitecture (non-hyper-threaded)	i386/haswell	8
Intel Haswell microarchitecture (hyper-threaded)	i386/haswell-ht	4
Intel Ivy Bridge microarchitecture (non-hyper-threaded)	i386/ivybridge	8
Intel Ivy Bridge microarchitecture (hyper-threaded)	i386/ivybridge-ht	4
Intel Sandy Bridge microarchitecture (non-hyper-threaded)	i386/sandybridge	8
Intel Sandy Bridge microarchitecture	i386/sandybridge-ht	4
TIMER_INT	timer	1

Use [Table 21.3, “OProfile Processors and Counters”](#) to verify that the correct processor type was detected and to determine the number of events that can be monitored simultaneously. **timer** is used as the processor type if the processor does not have supported performance monitoring hardware.

If **timer** is used, events cannot be set for any processor because the hardware does not have support for hardware performance counters. Instead, the timer interrupt is used for profiling.

If **timer** is not used as the processor type, the events monitored can be changed, and counter 0 for the processor is set to a time-based event by default. If more than one counter exists on the processor, the counters other than counter 0 are not set to an event by default. The default events monitored are shown in [Table 21.4, “Default Events”](#).

Table 21.4. Default Events

Processor	Default Event for Counter	Description
AMD Athlon and AMD64	CPU_CLK_UNHALTED	The processor's clock is not halted
AMD Family 10h, AMD Family 11h, AMD Family 12h	CPU_CLK_UNHALTED	The processor's clock is not halted
AMD Family 14h, AMD Family 15h	CPU_CLK_UNHALTED	The processor's clock is not halted
IBM POWER4	CYCLES	Processor Cycles
IBM POWER5	CYCLES	Processor Cycles
IBM PowerPC 970	CYCLES	Processor Cycles
Intel Core i7	CPU_CLK_UNHALTED	The processor's clock is not halted
Intel Nehalem microarchitecture	CPU_CLK_UNHALTED	The processor's clock is not halted
Intel Pentium 4 (hyper-threaded and non-hyper-threaded)	GLOBAL_POWER_EVENTS	The time during which the processor is not stopped
Intel Westmere microarchitecture	CPU_CLK_UNHALTED	The processor's clock is not halted
TIMER_INT	(none)	Sample for each timer interrupt

The number of events that can be monitored at one time is determined by the number of counters for the processor. However, it is not a one-to-one correlation; on some processors, certain events must be mapped to specific counters. To determine the number of counters available, execute the following command:

```
ls -d /dev/oprofile/[0-9]*
```

The events available vary depending on the processor type. To determine the events available for profiling, execute the following command as root (the list is specific to the system's processor type):

```
ophelp
```

Make sure that OProfile is configured

Unless OProfile is properly configured, the **ophelp** fails with the following error message:

```
Unable to open cpu_type file for reading
Make sure you have done opcontrol --init
cpu_type 'unset' is not valid
you should upgrade oprofile or force the use of timer mode
```

To configure OProfile, follow the instructions in [Section 21.3, “Configuring OProfile Using Legacy Mode”](#).

The events for each counter can be configured via the command line or with a graphical interface. For more information on the graphical interface, see [Section 21.10, “Graphical Interface”](#). If the counter cannot be set to a specific event, an error message is displayed.

To set the event for each configurable counter via the command line, use **opcontrol**:

```
opcontrol --event=event-name:sample-rate
```

Replace *event-name* with the exact name of the event from **ophelp**, and replace *sample-rate* with the number of events between samples.

21.3.2.1. Sampling Rate

By default, a time-based event set is selected. It creates a sample every 100,000 clock cycles per processor. If the timer interrupt is used, the timer is set to the respective rate and is not user-settable. If the **cpu_type** is not **timer**, each event can have a *sampling rate* set for it. The sampling rate is the number of events between each sample snapshot.

When setting the event for the counter, a sample rate can also be specified:

```
opcontrol --event=event-name:sample-rate
```

Replace *sample-rate* with the number of events to wait before sampling again. The smaller the count, the more frequent the samples. For events that do not happen frequently, a lower count may be needed to capture the event instances.



Sampling too frequently can overload the system

Be extremely careful when setting sampling rates. Sampling too frequently can overload the system, causing the system to appear as if it is frozen or causing the system to actually freeze.

21.3.2.2. Unit Masks

Some user performance monitoring events may also require unit masks to further define the event.

Unit masks for each event are listed with the **ophelp** command. The values for each unit mask are listed in hexadecimal format. To specify more than one unit mask, the hexadecimal values must be combined using a bitwise *or* operation.

```
opcontrol --event=event-name:sample-rate:unit-mask
```

21.3.3. Separating Kernel and User-space Profiles

By default, kernel mode and user mode information is gathered for each event. To configure OProfile to ignore events in kernel mode for a specific counter, execute the following command:

```
opcontrol --event=event-name:sample-rate:unit-mask:0
```

Execute the following command to start profiling kernel mode for the counter again:

```
opcontrol --event=event-name:sample-rate:unit-mask:1
```

To configure OProfile to ignore events in user mode for a specific counter, execute the following command:

```
opcontrol --event=event-name:sample-rate:unit-mask:1:0
```

Execute the following command to start profiling user mode for the counter again:

```
opcontrol --event=event-name:sample-rate:unit-mask:1:1
```

When the OProfile daemon writes the profile data to sample files, it can separate the kernel and library profile data into separate sample files. To configure how the daemon writes to sample files, execute the following command as root:

```
opcontrol --separate=choice
```

choice can be one of the following:

- ▶ **none** — Do not separate the profiles (default).
- ▶ **library** — Generate per-application profiles for libraries.
- ▶ **kernel** — Generate per-application profiles for the kernel and kernel modules.
- ▶ **all** — Generate per-application profiles for libraries and per-application profiles for the kernel and kernel modules.

If **--separate=library** is used, the sample file name includes the name of the executable as well as the name of the library.



Restart the OProfile profiler

These configuration changes will take effect when the OProfile profiler is restarted.

21.4. Starting and Stopping OProfile Using Legacy Mode

To start monitoring the system with OProfile, execute the following command as root:

```
opcontrol --start
```

Output similar to the following is displayed:

```
Using log file /var/lib/oprofile/oprofiled.log Daemon started. Profiler running.
```

The settings in **/root/.oprofile/daemonrc** are used.

The OProfile daemon, **oprofiled**, is started; it periodically writes the sample data to the **/var/lib/oprofile/samples/** directory. The log file for the daemon is located at **/var/lib/oprofile/oprofiled.log**.



Disable the nmi_watchdog registers

On a Red Hat Enterprise Linux 7 system, the **nmi_watchdog** registers with the **perf** subsystem. Due to this, the **perf** subsystem grabs control of the performance counter registers at boot time, blocking OProfile from working.

To resolve this, either boot with the **nmi_watchdog=0** kernel parameter set, or run the following command as **root** to disable **nmi_watchdog** at run time:

```
echo 0 > /proc/sys/kernel/nmi_watchdog
```

To re-enable **nmi_watchdog**, use the following command as **root**:

```
echo 1 > /proc/sys/kernel/nmi_watchdog
```

To stop the profiler, execute the following command as root:

```
opcontrol --shutdown
```

21.5. Saving Data in Legacy Mode

Sometimes it is useful to save samples at a specific time. For example, when profiling an executable, it may be useful to gather different samples based on different input data sets. If the number of events to be monitored exceeds the number of counters available for the processor, multiple runs of OProfile can be used to collect data, saving the sample data to different files each time.

To save the current set of sample files, execute the following command, replacing *name* with a unique descriptive name for the current session.

```
opcontrol --save=name
```

The directory `/var/lib/oprofile/samples/name/` is created and the current sample files are copied to it.

To specify the session directory to hold the sample data, use the `--session-dir`. If not specified, the data is saved in the `oprofile_data/` directory on the current path.

21.6. Analyzing the Data

The same OProfile post-processing tools are used whether you collect your profile with `operf` or `opcontrol` in legacy mode.

By default, `operf` stores the profiling data in the `current_dir/oprofile_data` directory. You can change to a different location with the `--session-dir` option. The usual post-profiling analysis tools such as `opreport` and `opannotate` can be used to generate profile reports. These tools search for samples in `current_dir/oprofile_data` first. If that directory does not exist, the analysis tools use the standard session directory of `/var/lib/oprofile/`. Statistics, such as total samples received and lost samples, are written to the `session_dir/samples/operf.log` file.

When using legacy mode, the OProfile daemon, `oprofiled`, periodically collects the samples and writes them to the `/var/lib/oprofile/samples/` directory. Before reading the data, make sure all data has been written to this directory by executing the following command as root:

```
opcontrol --dump
```

Each sample file name is based on the name of the executable. For example, the samples for the default event on a Pentium III processor for `/bin/bash` becomes:

```
\{root\}/bin/bash/\{dep\}/\{root\}/bin/bash/CPU_CLK_UNHALTED.100000
```

The following tools are available to profile the sample data once it has been collected:

- ▶ **opreport**
- ▶ **opannotate**

Use these tools, along with the binaries profiled, to generate reports that can be further analyzed.



Back up the executable and the sample files

The executable being profiled must be used with these tools to analyze the data. If it must change after the data is collected, back up the executable used to create the samples as well as the sample files. Please note that the sample file and the binary have to agree. Making a backup is not going to work if they do not match. `oparchive` can be used to address this problem.

Samples for each executable are written to a single sample file. Samples from each dynamically linked library are also written to a single sample file. While OProfile is running, if the executable being monitored changes and a sample file for the executable exists, the existing sample file is automatically deleted. Thus, if the existing sample file is needed, it must be backed up, along with the executable used to create it.

before replacing the executable with a new version. The OProfile analysis tools use the executable file that created the samples during analysis. If the executable changes the analysis tools will be unable to analyze the associated samples. See [Section 21.5, “Saving Data in Legacy Mode”](#) for details on how to back up the sample file.

21.6.1. Using oreport

The **opreport** tool provides an overview of all the executables being profiled. To view this information, type:

```
opreport
```

The following is part of a sample output:

```
Profiling through timer interrupt
TIMER:0|
samples|      %|
-----
25926 97.5212 no-vmlinux
359  1.3504 pi
65   0.2445 Xorg
62   0.2332 libvte.so.4.4.0
56   0.2106 libc-2.3.4.so
34   0.1279 libglib-2.0.so.0.400.7
19   0.0715 libXft.so.2.1.2
17   0.0639 bash
8    0.0301 ld-2.3.4.so
8    0.0301 libgdk-x11-2.0.so.0.400.13
6    0.0226 libgobject-2.0.so.0.400.7
5    0.0188 oprofiled
4    0.0150 libpthread-2.3.4.so
4    0.0150 libgtk-x11-2.0.so.0.400.13
3    0.0113 libXrender.so.1.2.2
3    0.0113 du
1    0.0038 libcrypto.so.0.9.7a
1    0.0038 libpam.so.0.77
1    0.0038 libtermcap.so.2.0.8
1    0.0038 libX11.so.6.2
1    0.0038 libgthread-2.0.so.0.400.7
1    0.0038 libwnck-1.so.4.9.0
```

Each executable is listed on its own line. The first column is the number of samples recorded for the executable. The second column is the percentage of samples relative to the total number of samples. The third column is the name of the executable.

See the **opreport** man page for a list of available command line options, such as the **-r** option used to sort the output from the executable with the smallest number of samples to the one with the largest number of samples. You can also use the **-t** or **--threshold** option to trim the output of **opcontrol**.

21.6.2. Using oreport on a Single Executable

To retrieve more detailed profiled information about a specific executable, use:

```
opreport mode executable
```

Replace *executable* with the full path to the executable to be analyzed. *mode* stands for one of the following options:

-l

This option is used to list sample data by symbols. For example, running this command:

```
~]# opreport -l /lib/tls/libc-version.so
```

produces the following output:

```
samples % symbol name
12 21.4286 __gconv_transform_utf8_internal
5 8.9286 __int_malloc 4 7.1429 malloc
3 5.3571 __i686.get_pc_thunk.bx
3 5.3571 __dl_mcount_wrapper_check
3 5.3571 mbrtowc
3 5.3571 memcpy
2 3.5714 __int_realloc
2 3.5714 __nl_intern_locale_data
2 3.5714 free
2 3.5714 strcmp
1 1.7857 __ctype_get_mb_cur_max
1 1.7857 __unregister_atfork
1 1.7857 __write_nocancel
1 1.7857 __dl_addr
1 1.7857 __int_free
1 1.7857 __itoa_word
1 1.7857 calc_eclosure_iter
1 1.7857 fopen@@GLIBC_2.1
1 1.7857 getpid
1 1.7857 memmove
1 1.7857 msort_with_tmp
1 1.7857 strcpy
1 1.7857 strlen
1 1.7857 vfprintf
1 1.7857 write
```

The first column is the number of samples for the symbol, the second column is the percentage of samples for this symbol relative to the overall samples for the executable, and the third column is the symbol name.

To sort the output from the largest number of samples to the smallest (reverse order), use **-r** in conjunction with the **-l** option.

-i symbol-name

List sample data specific to a symbol name. For example, running:

```
~]# opreport -l -i __gconv_transform_utf8_internal
/lib/tls/libc-version.so
```

returns the following output:

```
samples % symbol name
12 100.000 __gconv_transform_utf8_internal
```

The first line is a summary for the symbol/executable combination.

The first column is the number of samples for the memory symbol. The second column is the percentage of samples for the memory address relative to the total number of samples for the symbol. The third column is the symbol name.

-d

This lists sample data by symbols with more detail than the **-l** option. For example, with the following command:

```
~]# opreport -d -i __gconv_transform_utf8_internal
/lib/tls/libc-version.so
```

this output is returned:

```
vma samples % symbol name
00a98640 12 100.000 __gconv_transform_utf8_internal
00a98640 1 8.3333
00a9868c 2 16.6667
00a9869a 1 8.3333
00a986c1 1 8.3333
00a98720 1 8.3333
00a98749 1 8.3333
00a98753 1 8.3333
00a98789 1 8.3333
00a98864 1 8.3333
00a98869 1 8.3333
00a98b08 1 8.3333
```

The data is the same as the **-l** option except that for each symbol, each virtual memory address used is shown. For each virtual memory address, the number of samples and percentage of samples relative to the number of samples for the symbol is displayed.

-e symbol-name...

With this option, you can exclude some symbols from the output. Replace *symbol-name* with the comma-separated list of symbols you want to exclude

session:name

Here, you can specify the full path to the session, a directory relative to the `/var/lib/oprofile/samples/` directory, or if you are using `oprf`, a directory relative to `./oprofile_data/samples/`.

21.6.3. Getting more detailed output on the modules

OProfile collects data on a system-wide basis for kernel- and user-space code running on the machine. However, once a module is loaded into the kernel, the information about the origin of the kernel module is lost. The module could have come from the `initrd` file on boot up, the directory with the various kernel modules, or a locally created kernel module. As a result, when OProfile records samples for a module, it just lists the samples for the modules for an executable in the root directory, but this is unlikely to be the place with the actual code for the module. You will need to take some steps to make sure that analysis tools get the proper executable.

To get a more detailed view of the actions of the module, you will need to either have the module "unstripped" (that is installed from a custom build) or have the `debuginfo` package installed for the kernel.

Find out which kernel is running with the `uname -a` command, obtain the appropriate `debuginfo` package and install it on the machine.

Then proceed with clearing out the samples from previous runs with the following command:

```
opcontrol --reset
```

To start the monitoring process, for example, on a machine with Westmere processor, run the following command:

```
~]# opcontrol --setup --vmlinuz=/usr/lib/debug/lib/modules/`uname -r`/vmlinuz - -event=CPU_CLK_UNHALTED:500000
```

Then the detailed information, for instance, for the ext4 module can be obtained with:

```
~]# opreport /ext4 -l --image-path /lib/modules/`uname -r`/kernel
CPU: Intel Westmere microarchitecture, speed 2.667e+06 MHz (estimated)
Counted CPU_CLK_UNHALTED events (Clock cycles when not halted) with a unit mask of
0x00 (No unit mask) count 500000
warning: could not check that the binary file /lib/modules/2.6.32-
191.el6.x86_64/kernel/fs/ext4/ext4.ko has not been modified since the profile was
taken. Results may be inaccurate.
samples % symbol name
1622 9.8381 ext4_iget
1591 9.6500 ext4_find_entry
1231 7.4665 __ext4_get_inode_loc
783 4.7492 ext4_ext_get_blocks
752 4.5612 ext4_check_dir_entry
644 3.9061 ext4_mark_iloc_dirty
583 3.5361 ext4_get_blocks
583 3.5361 ext4_xattr_get
479 2.9053 ext4_htree_store_dirent
469 2.8447 ext4_get_group_desc
414 2.5111 ext4_dx_find_entry
```

21.6.4. Using opannotate

The **opannotate** tool tries to match the samples for particular instructions to the corresponding lines in the source code. The resulting files generated should have the samples for the lines at the left. It also puts in a comment at the beginning of each function listing the total samples for the function.

For this utility to work, the appropriate *debuginfo* package for the executable must be installed on the system. On Red Hat Enterprise Linux, the *debuginfo* packages are not automatically installed with the corresponding packages that contain the executable. You have to obtain and install them separately.

The general syntax for **opannotate** is as follows:

```
opannotate --search-dirs src-dir --source executable
```

These command line options are mandatory. Replace *src-dir* with a path to the directory containing the source code and specify the executable to be analyzed. See the **opannotate** man page for a list of additional command line options.

21.7. Understanding /dev/oprofile/

When using OProfile in legacy mode, the **/dev/oprofile/** directory is used to store the file system for OProfile. On the other hand, the **operf** does not require **/dev/oprofile/**. Use the **cat** command to display the values of the virtual files in this file system. For example, the following command displays the type of processor OProfile detected:

```
cat /dev/oprofile/cpu_type
```

A directory exists in `/dev/oprofile/` for each counter. For example, if there are 2 counters, the directories `/dev/oprofile/0/` and `/dev/oprofile/1/` exist.

Each directory for a counter contains the following files:

- ▶ **count** — The interval between samples.
- ▶ **enabled** — If 0, the counter is off and no samples are collected for it; if 1, the counter is on and samples are being collected for it.
- ▶ **event** — The event to monitor.
- ▶ **extra** — Used on machines with Nehalem processors to further specify the event to monitor.
- ▶ **kernel** — If 0, samples are not collected for this counter event when the processor is in kernel-space; if 1, samples are collected even if the processor is in kernel-space.
- ▶ **unit_mask** — Defines which unit masks are enabled for the counter.
- ▶ **user** — If 0, samples are not collected for the counter event when the processor is in user-space; if 1, samples are collected even if the processor is in user-space.

The values of these files can be retrieved with the `cat` command. For example:

```
cat /dev/oprofile/0/count
```

21.8. Example Usage

While OProfile can be used by developers to analyze application performance, it can also be used by system administrators to perform system analysis. For example:

- ▶ *Determine which applications and services are used the most on a system* — `opreport` can be used to determine how much processor time an application or service uses. If the system is used for multiple services but is underperforming, the services consuming the most processor time can be moved to dedicated systems.
- ▶ *Determine processor usage* — The `CPU_CLK_UNHALTED` event can be monitored to determine the processor load over a given period of time. This data can then be used to determine if additional processors or a faster processor might improve system performance.

21.9. OProfile Support for Java

OProfile allows you to profile dynamically compiled code (also known as "just-in-time" or JIT code) of the Java Virtual Machine (JVM). OProfile in Red Hat Enterprise Linux 7 includes built-in support for the JVM Tools Interface (JVMTI) agent library, which supports Java 1.5 and higher.

21.9.1. Profiling Java Code

To profile JIT code from the Java Virtual Machine with the JVMTI agent, add the following to the JVM startup parameters:

```
-agentlib:jvmti_oprofile
```

Where `jvmti_oprofile` is a path to the oprofile agent. For 64-bit JVM, the path looks as follows:

```
-agentlib:/usr/lib64/oprofile/libjvmti_oprofile.so
```

Currently, you can add one command-line option: **--debug**, which enables debugging mode.

Install the oprofile-jit package

The *oprofile-jit* package must be installed on the system in order to profile JIT code with OProfile. With this package, you gain capability to show method-level information.

Depending on the JVM that you are using, you may have to install the *debuginfo* package for the JVM. For OpenJDK, this package is required, there is no *debuginfo* package for Oracle JDK. To keep the debug information packages synchronized with their respective non-debug packages, you also need to install the *yum-plugin-auto-update-debug-info* plug-in. This plug-in searches the debug information repository for corresponding updates.

After successful setup, you can apply the standard profiling and analyzing tools described in previous sections

To learn more about Java support in OProfile, see the OProfile Manual, which is linked from [Section 21.12, “Additional Resources”](#).

21.10. Graphical Interface

Some OProfile preferences can be set with a graphical interface. To start it, execute the **oprof_start** command as root at a shell prompt. To use the graphical interface, you will need to have the **oprofile-gui** package installed.

After changing any of the options, save them by clicking the **Save and quit** button. The preferences are written to **/root/.oprofile/daemonrc**, and the application exits.

Clicking the Save and quit button

Exiting the application does not stop OProfile from sampling.

On the **Setup** tab, to set events for the processor counters as discussed in [Section 21.3.2, “Setting Events to Monitor”](#), select the counter from the pulldown menu and select the event from the list. A brief description of the event appears in the text box below the list. Only events available for the specific counter and the specific architecture are displayed. The interface also displays whether the profiler is running and some brief statistics about it.

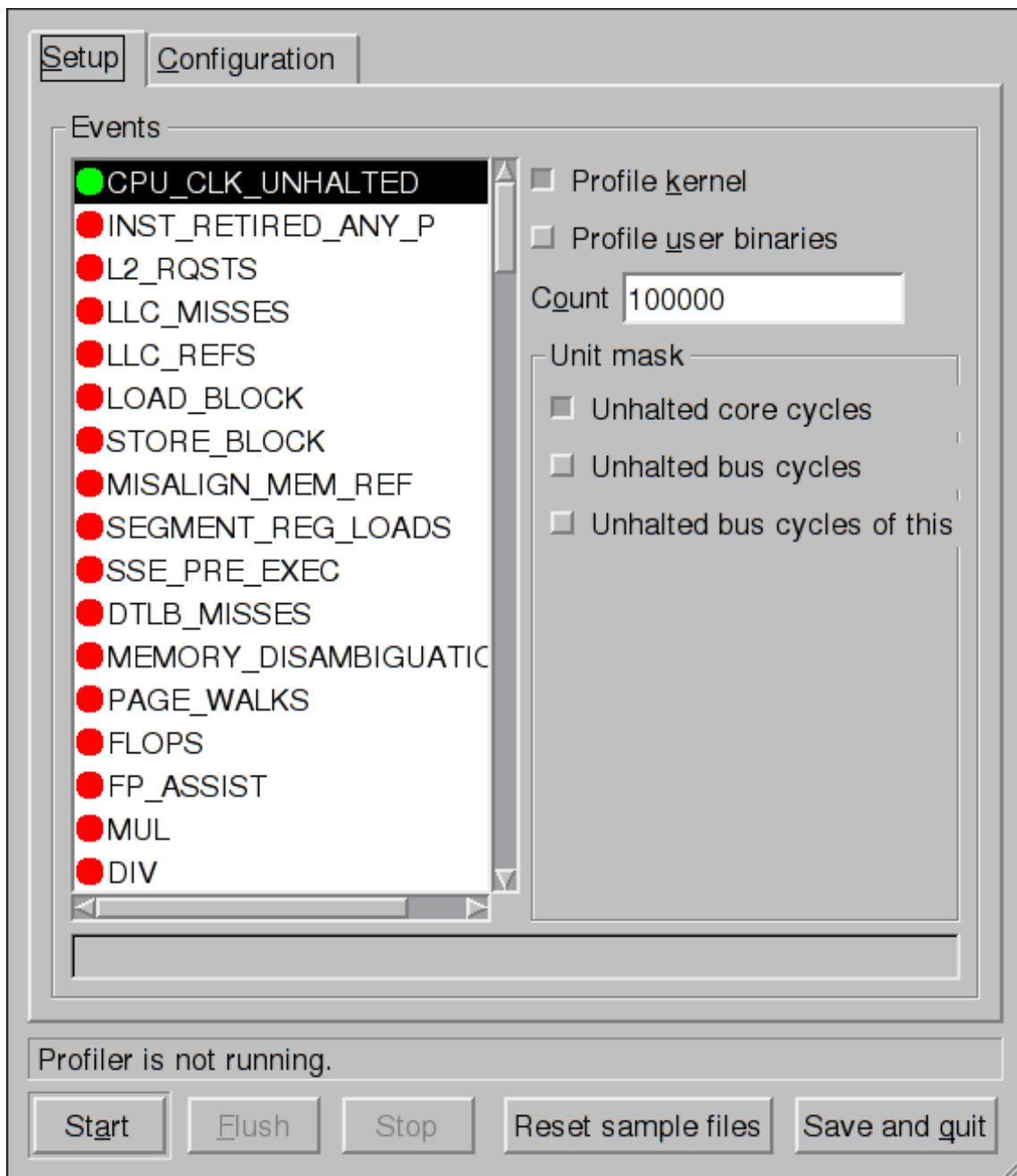


Figure 21.1. OProfile Setup

On the right side of the tab, select the **Profile kernel** option to count events in kernel mode for the currently selected event, as discussed in [Section 21.3.3, “Separating Kernel and User-space Profiles”](#). If this option is not selected, no samples are collected for the kernel.

Select the **Profile user binaries** option to count events in user mode for the currently selected event, as discussed in [Section 21.3.3, “Separating Kernel and User-space Profiles”](#). If this option is not selected, no samples are collected for user applications.

Use the **Count** text field to set the sampling rate for the currently selected event as discussed in [Section 21.3.2.1, “Sampling Rate”](#).

If any unit masks are available for the currently selected event, as discussed in [Section 21.3.2.2, “Unit Masks”](#), they are displayed in the **Unit Masks** area on the right side of the **Setup** tab. Select the checkbox beside the unit mask to enable it for the event.

On the **Configuration** tab, to profile the kernel, enter the name and location of the **linux** file for the kernel to monitor in the **Kernel image file** text field. To configure OProfile not to monitor the kernel, select **No kernel image**.

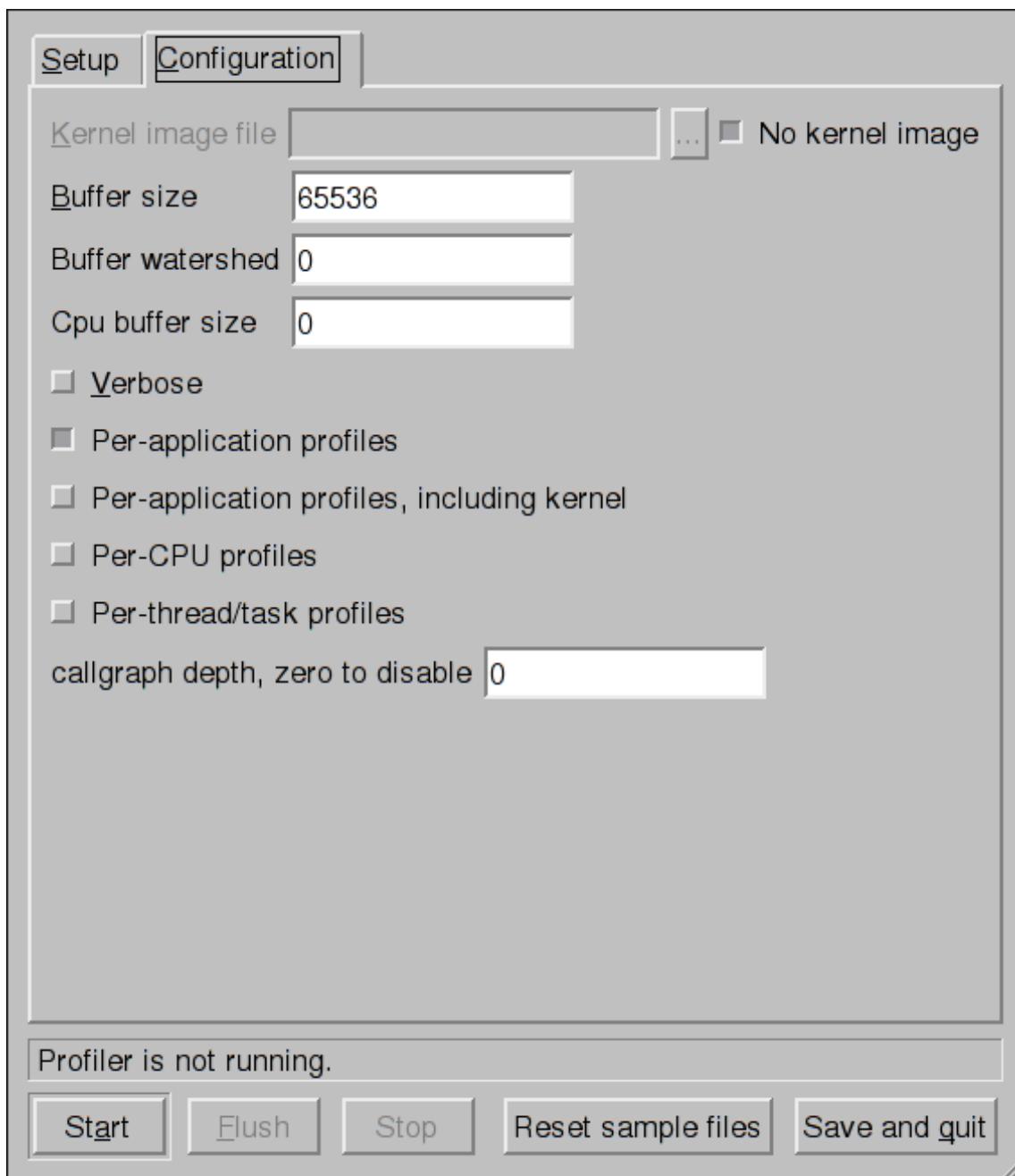


Figure 21.2. OProfile Configuration

If the **Verbose** option is selected, the **oprofiled** daemon log includes more information.

If **Per-application profiles** is selected, OProfile generates per-application profiles for libraries. This is equivalent to the **opcontrol --separate=library** command. If **Per-application profiles, including kernel** is selected, OProfile generates per-application profiles for the kernel and kernel modules as discussed in [Section 21.3.3, “Separating Kernel and User-space Profiles”](#). This is equivalent to the **opcontrol --separate=kernel** command.

To force data to be written to samples files as discussed in [Section 21.6, “Analyzing the Data”](#), click the **Flush** button. This is equivalent to the **opcontrol --dump** command.

To start OProfile from the graphical interface, click **Start**. To stop the profiler, click **Stop**. Exiting the application does not stop OProfile from sampling.

21.11. OProfile and SystemTap

SystemTap is a tracing and probing tool that allows users to study and monitor the activities of the operating system in fine detail. It provides information similar to the output of tools like **netstat**, **ps**, **top**, and **iostat**; however, SystemTap is designed to provide more filtering and analysis options for collected information.

While using OProfile is suggested in cases of collecting data on where and why the processor spends time in a particular area of code, it is less usable when finding out why the processor stays idle.

You might want to use SystemTap when instrumenting specific places in code. Because SystemTap allows you to run the code instrumentation without having to stop and restart the instrumented code, it is particularly useful for instrumenting the kernel and daemons.

For more information on SystemTap, see [Section 21.12.2, “Useful Websites”](#) for the relevant SystemTap documentation.

21.12. Additional Resources

This chapter only highlights OProfile and how to configure and use it. To learn more, see the following resources.

21.12.1. Installed Docs

- ▶ `/usr/share/doc/oprofile-version/oprofile.html` — *OProfile Manual*
- ▶ `oprofile` man page — Discusses **opcontrol**, **opreport**, **opannotate**, and **ophelp**
- ▶ `operf` man page

21.12.2. Useful Websites

- ▶ <http://oprofile.sourceforge.net/> — Contains the latest documentation, mailing lists, IRC channels, and more.
- ▶ [SystemTap Beginners Guide](#) — Provides basic instructions on how to use SystemTap to monitor different subsystems of Red Hat Enterprise Linux in finer detail.

Part VI. Kernel, Module and Driver Configuration

This part covers various tools that assist administrators with kernel customization.

Chapter 22. Working with the GRUB 2 Boot Loader

Red Hat Enterprise Linux 7 is distributed with the GNU GRand Unified Boot loader (GRUB) version 2 boot loader, which allows the user to select an operating system or kernel to be loaded at system boot time. GRUB 2 also allows the user to pass arguments to the kernel.

22.1. Configuring the GRUB 2 Boot Loader

GRUB 2 reads its configuration from the **/boot/grub2/grub.cfg** file, or **/boot/efi/EFI/redhat/grub.cfg** on UEFI machines. This file contains menu information, however, it is not supposed to be edited as it is generated by the **/usr/sbin/grub2-mkconfig** utility based on Linux kernels located in the **/boot/** directory, template files located in **/etc/grub.d/**, and custom settings in the **/etc/default/grub** file. Any manual edits could therefore cause the changes to be lost during updates. The **grub.cfg** file is automatically updated each time a new kernel is installed. To update this configuration file manually, type the following at a shell prompt as **root**:

```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

Alternatively, on UEFI systems, run the following:

```
~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

Among various code snippets and directives, the **grub.cfg** configuration file contains one or more **menuentry** blocks, each representing a single GRUB 2 boot menu entry. These blocks always start with the **menuentry** keyword followed by a title, list of options, and an opening curly bracket, and end with a closing curly bracket. Anything between the opening and closing bracket should be indented. For example, the following is a sample **menuentry** block for Red Hat Enterprise Linux 7 with Linux kernel 3.8.0-0.40.el7.x86_64:

```
menuentry 'Red Hat Enterprise Linux Client' --class red --class gnu-linux --class gnu
--class os $menuentry_id_option 'gnulinux-simple-c60731dc-9046-4000-9182-
64bdcce08616' {
    load_video
    set gfxpayload=keep
    insmod gzio
    insmod part_msdos
    insmod xfs
    set root='hd0,msdos1'
    if [ x$feature_platform_search_hint = xy ]; then
        search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-
efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 --hint='hd0,msdos1' 19d9e294-65f8-
4e37-8e73-d41d6daa6e58
    else
        search --no-floppy --fs-uuid --set=root 19d9e294-65f8-4e37-8e73-
d41d6daa6e58
    fi
    echo      'Loading Linux 3.8.0-0.40.el7.x86_64 ...'
    linux    /vmlinuz-3.8.0-0.40.el7.x86_64 root=/dev/mapper/rhel-root ro
    rd.md=0 rd.dm=0 rd.lvm.lv=rhel/swap crashkernel=auto rd.luks=0 vconsole.keymap=us
    rd.lvm.lv=rhel/root rhgb quiet
    echo      'Loading initial ramdisk ...'
    initrd  /initramfs-3.8.0-0.40.el7.x86_64.img
}
```

Each **menuentry** block that represents an installed Linux kernel contains **linux** (**linuxefi** on UEFI systems) and **initrd** directives followed by the path to the kernel and the **initramfs** image respectively. If a separate **/boot** partition was created, the paths to the kernel and the **initramfs** image are relative to **/boot**. In the example above, the **initrd /initramfs-3.8.0-0.40.el7.x86_64.img** line means that the **initramfs** image is actually located at **/boot/initramfs-3.8.0-0.40.el7.x86_64.img** when the root file system is mounted, and likewise for the kernel path.

The kernel version number as given on the **linux /mlinuz-kernel_version** line must match the version number of the **initramfs** image given on the **initrd /initramfs-kernel_version.img** line of each **menuentry** block. For more information on how to verify the initial RAM disk image, see [Section 23.5, “Verifying the Initial RAM Disk Image”](#).

The initrd directive in grub.cfg refers to an initramfs image

In **menuentry** blocks, the **initrd** directive must point to the location (relative to the **/boot** directory if it is on a separate partition) of the **initramfs** file corresponding to the same kernel version. This directive is called **initrd** because the previous tool which created initial RAM disk images, **mkinitrd**, created what were known as **initrd** files. The **grub.cfg** directive remains **initrd** to maintain compatibility with other tools. The file-naming convention of systems using the **dracut** utility to create the initial RAM disk image is **initramfs-kernel_version.img**.

For information on using **Dracut**, see [Section 23.5, “Verifying the Initial RAM Disk Image”](#).

22.2. Customizing GRUB 2 Menu

GRUB 2 scripts search the user's computer and build a boot menu based on what operating systems the scripts find. To reflect the latest system boot options, the boot menu is rebuilt automatically when the kernel is updated or a new kernel is added.

However, users may want to build a menu containing specific entries or to have the entries in a specific order. GRUB 2 allows basic customization of the boot menu to give users control of what actually appears on the screen.

GRUB 2 uses a series of scripts to build the menu; these are located in the **/etc/grub.d/** directory and include:

- ▶ **00_header**, which loads GRUB 2 settings from the **/etc/default/grub** file.
- ▶ **10_linux**, which locates kernels in the default partition of Red Hat Enterprise Linux.
- ▶ **30_os-prober**, which builds entries for operating systems found on other partitions.
- ▶ **40_custom**, a template, which can be used to create additional menu entries.

Scripts from the **/etc/grub.d/** directory are read in alphabetical order and can be therefore renamed to change the boot order of specific menu entries.



Causing the GRUB 2 boot menu to display

If you set the **GRUB_TIMEOUT** key in the **/etc/default/grub** file to **0**, GRUB 2 will not display its list of bootable kernels when the system starts up. In order to display this list when booting, press and hold any alphanumeric key while and immediately after BIOS information is displayed; GRUB 2 will present you with the GRUB menu.

22.2.1. Changing the Default Boot Entry

By default, the **saved** value is used for the **GRUB_DEFAULT** key in the **/etc/default/grub** file. This instructs GRUB 2 to load the kernel specified by the **saved_entry** directive in the GRUB 2 environment file, **/boot/grub2/grubenv**. You can set another GRUB record to be the default, using the **grub2-set-default** command, which will update the GRUB 2 environment file. The **saved_entry** default value is the name of last kernel to be installed.

GRUB 2 supports using a numeric value as the key for the **saved_entry** directive to change the default order in which the operating systems are loaded. To specify which operating system should be loaded first, use the **grub2-set-default** command as follows:

```
~]# grub2-set-default 2
```

Note that the position of a menu entry in the list is denoted by a number starting with zero; therefore, in the example above, the third entry will be loaded. This value will be over-written by the name of the next kernel to be installed.

To fix a system to always use a particular kernel, use its name as the key to the **GRUB_DEFAULT** directive in the **/etc/default/grub** file. To list the names, run the following command as **root**:

```
~]# awk -F\ ' '$1=="menuentry " {print $2}' /etc/grub2.cfg
```

Note that any changes to **/etc/default/grub** require rebuilding the **grub.cfg** file. Run the **grub2-mkconfig -o /boot/grub2/grub.cfg** or, on UEFI systems, **grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg** command to rebuild and replace the configuration file.

22.2.2. Editing an Entry

Kernel Parameters

To use a kernel parameter only during a single boot process, when the GRUB 2 boot menu appears, move the cursor to the kernel you want to start, press the **e** key to edit the line with the kernel and add the kernel parameter. For example, to run the system in emergency mode, add the **emergency** parameter at the end of the **linux** line:

```
linux      /vmlinuz-3.10.0-0.rc4.59.el7.x86_64 root=/dev/mapper/rhel-root ro
rd.md=0 rd.dm=0 rd.lvm.lv=rhel/swap crashkernel=auto rd.luks=0 vconsole.keymap=us
rd.lvm.lv=rhel/root rhgb quiet emergency
```

These settings are, however, not persistent and apply only for a single boot. To make the settings persistent, edit values of the **GRUB_CMDLINE_LINUX** key in the **/etc/default/grub** file. For example, if you want to enable emergency mode for each boot, edit the entry as follows:

```
GRUB_CMDLINE_LINUX="emergency"
```

Note that you can specify multiple parameters for the **GRUB_CMDLINE_LINUX** key, similarly to adding the parameters in the GRUB 2 boot menu. For example,

```
console=tty0 console=ttyS0,9600n8
```

Where **console=tty0** is the first virtual terminal and **console=ttyS0** is the serial terminal to be used.

Then, run the **grub2-mkconfig -o /boot/grub2/grub.cfg** command, or **grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg** on UEFI systems, to update the configuration file.

22.2.3. Adding a new Entry

When executing the **grub2-mkconfig** command, GRUB 2 searches for Linux kernels and other operating systems based on the files located in the **/etc/grub.d/** directory. The **10_linux** script searches for installed Linux kernels on the same partition. The **30_os-prober** searches for other operating systems. Menu entries are also automatically added to the boot menu when updating the kernel.

The **40_custom** file located in the **/etc/grub.d/** directory is a template for custom entries and looks as follows:

```
#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries. Simply type the
# menu entries you want to add after this comment. Be careful not to change
# the 'exec tail' line above.
```

This file can be edited or copied. Note that as a minimum, a valid menu entry must include at least the following:

```
menuentry "<Title>"{
<Data>
}
```

22.2.4. Using only a Custom Menu

If you do not wish menu entries to be updated automatically, you can create a custom menu.



Backup of /etc/grub.d/

Before proceeding, back up the contents of the **/etc/grub.d/** directory in case you need to revert the changes later.



Note

Note that modifying the **/etc/default/grub** file does not have any effect on creating custom menus.

1. Copy and paste the contents of the **/boot/grub2/grub.cfg** or **/boot/efi/EFI/redhat/grub.cfg** file in the **/etc/grub.d/40_custom** file below the existing header lines; the executable part of the **40_custom** script has to be preserved.
2. Remove lines above the first menu entry except the existing header lines above.

This is an example of a custom **40_custom** script:

```
#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries. Simply type the
# menu entries you want to add after this comment. Be careful not to change
# the 'exec tail' line above.

menuentry 'First custom entry' --class red --class gnu-linux --class gnu --class
os ${menuentry_id_option} 'gnulinux-3.10.0-67.el7.x86_64-advanced-32782dd0-4b47-
4d56-a740-2076ab5e5976' {
    load_video
    set gfxpayload=keep
    insmod gzio
    insmod part_msdos
    insmod xfs
    set root='hd0,msdos1'
    if [ ${feature_platform_search_hint} = xy ]; then
        search --no-floppy --fs-uuid --set=root --hint='hd0,msdos1'
7885bba1-8aa7-4e5d-a7ad-821f4f52170a
    else
        search --no-floppy --fs-uuid --set=root 7885bba1-8aa7-4e5d-a7ad-
821f4f52170a
    fi
    linux16 /vmlinuz-3.10.0-67.el7.x86_64 root=/dev/mapper/rhel-root ro
rd.lvm.lv=rhel/root vconsole.font=latarcyrheb-sun16 rd.lvm.lv=rhel/swap
vconsole.keymap=us crashkernel=auto rhgb quiet LANG=en_US.UTF-8
        initrd16 /initramfs-3.10.0-67.el7.x86_64.img
}
menuentry 'Second custom entry' --class red --class gnu-linux --class gnu --
class os ${menuentry_id_option} 'gnulinux-0-rescue-
07f43f20a54c4ce8ada8b70d33fd001c-advanced-32782dd0-4b47-4d56-a740-
2076ab5e5976' {
    load_video
    insmod gzio
    insmod part_msdos
    insmod xfs
    set root='hd0,msdos1'
    if [ ${feature_platform_search_hint} = xy ]; then
        search --no-floppy --fs-uuid --set=root --hint='hd0,msdos1'
7885bba1-8aa7-4e5d-a7ad-821f4f52170a
    else
        search --no-floppy --fs-uuid --set=root 7885bba1-8aa7-4e5d-a7ad-
821f4f52170a
    fi
    linux16 /vmlinuz-0-rescue-07f43f20a54c4ce8ada8b70d33fd001c
root=/dev/mapper/rhel-root ro rd.lvm.lv=rhel/root vconsole.font=latarcyrheb-
sun16 rd.lvm.lv=rhel/swap vconsole.keymap=us crashkernel=auto rhgb quiet
        initrd16 /initramfs-0-rescue-07f43f20a54c4ce8ada8b70d33fd001c.img
}
```

3. Remove all files from the **/etc/grub.d** directory except the following:

- » **00_header**,
- » **40_custom**,
- » and **README**.

4. Edit, add, or remove menu entries in the **40_custom** file as desired.

Alternatively, if you wish to keep the files in the `/etc/grub2.d/` directory, make them unexecutable by running the `chmod a-x <file_name>` command.

5. Update the `grub.cfg` file by running the `grub2-mkconfig -o /boot/grub2/grub.cfg` command or `grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg` on UEFI systems.

22.3. GRUB 2 Password Protection

GRUB 2 supports basic password protection that uses unencrypted passwords in the GRUB 2 template files. To enable this functionality, specify a superuser, who can reach the protected entries, and the password. Other users can be specified to access these entries as well. To specify which menu entries should be password-protected, edit the `/etc/grub.d/00_header` file. Alternatively, if you wish to preserve the settings after GRUB 2 upgrades, modify the `/etc/grub.d/40_custom` file.

22.3.1. Setting Up Users and Password Protection, Specifying Menu Entries

1. To specify a superuser, add the following lines in the `/etc/grub.d/00_header` file, where `john` is the name of the user designated as the superuser, and `johnspassword` is the superuser's password:

```
cat <<EOF
set superusers="john"
password john johnspassword
EOF
```

2. To allow other users to access the menu entries, add additional lines per user at the end of the `/etc/grub.d/00_header` file.

```
cat <<EOF
set superusers="john"
password john johnspassword
password jane janespword
EOF
```

3. When the users and passwords are set up, specify the menu entries that should be password-protected. If you do not specify any menu entries, all menu entries will be password-protected by default. See [Section 22.2.3, “Adding a new Entry”](#) for the format of menu entries and see [Section 22.3.2, “Preserving the Setup after GRUB 2 Updates”](#) for an example of specifying menu entries to be protected.
4. Run the `grub2-mkconfig -o /boot/grub2/grub.cfg` command to update the configuration file. Or the `grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg` command on UEFI systems.

22.3.2. Preserving the Setup after GRUB 2 Updates

Alternatively, if you wish to preserve the changes after future GRUB 2 updates, add lines in the `/etc/grub.d/40_custom` file in a similar fashion:

```
set superusers="john"
password john johnspassword
password jane janespword

menuentry 'Red Hat Enterprise Linux Client' {
    set root=(hd0,msdos1)
```

```
linux /vmlinuz
}

menuentry 'Fedora' --users jane {
set root=(hd0,msdos2)
linux /vmlinuz
}
```

In the above example, **john** is the **superuser** and can therefore boot any menu entry, use the GRUB 2 command line, and edit items of the GRUB 2 menu during boot. In this case, **john** can access both Red Hat Enterprise Linux Client and Fedora. Anyone can boot Red Hat Enterprise Linux Client. User **jane** can boot Fedora since she was granted the permission in the configuration. If you do not specify a menu entry, the password protection function will not work.

After you have made the changes in the template file, run the following command to update the GRUB 2 configuration file:

```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

Alternatively, on UEFI systems, run the following:

```
~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

Even if you do not specify a user and password for a menu entry, the superuser's password will be asked for when accessing such a system.

22.3.3. Password Encryption

By default, passwords are saved in plain text in GRUB 2 scripts. Although the files cannot be accessed on boot without the correct password, security can be improved by encrypting the password using the **grub2-mkpasswd-pbkdf2** command. This command converts a desired password into a long hash, which is placed in the GRUB 2 scripts instead of the plain-text password.

1. To generate an encrypted password, run the **grub2-mkpasswd-pbkdf2** command on the command line as **root**.
2. Enter the desired password when prompted and repeat it. The command then outputs your password in an encrypted form.
3. Copy the hash, and paste it in the template file where you configured the users, that is, either in **/etc/grub.d/00_header** or **/etc/grub.d/40_custom**. The following format applies for the **00_header** file:

```
cat <<EOF
setsuperusers="john"
password_pbkdf2 john
grub.pbkdf2.sha512.10000.19074739ED80F115963D984BDCB35AA671C24325755377C3E9B01
4D862DA6ACC77BC110EED41822800A87FD3700C037320E51E9326188D53247EC0722DDF15FC.C
56EC0738911AD86CEA55546139FEB366A393DF9785A8F44D3E51BF09DB980BAFEF85281CBBC5
6778D8B19DC94833EA8342F7D73E3A1AA30B205091F1015A85
EOF
```

The following format applies for the **40_header** file:

```
setsuperusers="john"
password_pbkdf2 john
grub.pbkdf2.sha512.10000.19074739ED80F115963D984BDCB35AA671C24325755377C3E9B01
```

```
4D862DA6ACC77BC110EED41822800A87FD3700C037320E51E9326188D53247EC0722DDF15FC.C
56EC0738911AD86CEA55546139FEB366A393DF9785A8F44D3E51BF09DB980BAFEF85281CBBC5
6778D8B19DC94833EA8342F7D73E3A1AA30B205091F1015A85
```



Warning

If you do not use the correct format, or modify the configuration in an incorrect way, you might be unable to boot your system.

22.4. Reinstalling GRUB 2

Reinstalling GRUB 2 might be a useful and easier way to fix certain problems usually caused by an incorrect installation of GRUB 2, missing files, or a broken system. Other reasons to reinstall GRUB 2 include the following:

- ▶ Upgrading from the previous version of GRUB.
- ▶ The user requires the GRUB 2 boot loader to control installed operating systems. However, some operating systems are installed with their own boot loaders. Reinstalling GRUB 2 returns control to the desired operating system.
- ▶ Adding the boot information to another drive.

22.4.1. Using the `grub2-install` Command

When using the `grub2-install` command, the boot information is updated and missing files are restored. Note that the files are restored only if they are not corrupted. If the `/boot/grub2/` directory is missing, it will be recreated.

Use the `grub2-install <device>` command to reinstall GRUB 2 if the system is operating normally. For example:

```
~]# grub2-install /dev/sda
```

22.4.2. Resetting and Reinstalling GRUB 2

This method completely removes all GRUB 2 configuration files and system settings, and is therefore used when the user wishes to reset all configuration setting to the default values. Removing of the configuration files and subsequent reinstalling of GRUB 2 fixes failures caused by corrupted files and incorrect configuration. To do so, as `root`, follow these steps:

1. Run the `rm /etc/grub.d/*` command;
2. Run the `rm /etc/sysconfig/grub` command;
3. Run the `yum reinstall grub2-tools` command;
4. Run the `grub2-mkconfig -o /boot/grub2/grub.cfg` or, on UEFI systems, `grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg` command to replace the configuration file;
5. Now follow the procedure in [Section 22.4.1, “Using the `grub2-install` Command”](#) to restore GRUB2 on the `/boot/` partition.

22.5. GRUB 2 over Serial Console

If you use computers with no display or keyboard, it can be very useful to control the machines through serial communications.

22.5.1. Configuring GRUB 2

In order to use GRUB 2 over a serial line, add the following two lines in the `/etc/default/grub` file:

```
GRUB_TERMINAL="serial"
GRUB_SERIAL_COMMAND="serial --speed=9600 --unit=0 --word=8 --parity=no --stop=1"
```

The first line disables the graphical terminal. Note that specifying the `GRUB_TERMINAL` key overrides values of `GRUB_TERMINAL_INPUT` and `GRUB_TERMINAL_OUTPUT`. On the second line, adjust the baud rate, parity, and other values to fit your environment and hardware. A much higher baud rate, for example **115200**, is preferable for tasks such as following log files.

Once you have completed the changes in the `/etc/default/grub` file, it is necessary to execute the `grub2-mkconfig -o /boot/grub2/grub.cfg` or `grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg` command to update the GRUB 2 configuration file.

 **Note**

This section explains configuring the grub terminal for access over a serial connection but an additional option must be added to the kernel definition to make that kernel monitor a serial connection. For example:

```
console=ttyS0,9600n8
```

Where `console=ttyS0` is the serial terminal to be used, **9600** is the baud rate, **n** is for no parity, and **8** is the word length in bits. For more information on adding kernel options, see [Section 22.2.2, "Editing an Entry"](#). For more information on serial console settings, see <https://www.kernel.org/doc/Documentation/serial-console.txt>

22.5.2. Using screen to Connect to the Serial Console

The `screen` tool serves as a capable serial terminal. To install it, run as `root`:

```
~]# yum install screen
```

To connect to your machine using the serial console, run the following command:

```
~]$ screen /dev/<console_port>
```

By default, if no option is specified, `screen` uses the standard 9600 baud rate. To set a different baud rate, run:

```
~]$ screen /dev/<console_port> 115200
```

To end the session in `screen`, press `Ctrl+a`, type `:quit` and press `Enter`.

See the `screen(1)` manual page for additional options and detailed information.

22.6. Terminal Menu Editing During Boot

Menu entries can be modified and arguments passed to the kernel on boot. This is done using the menu entry editor interface, which is triggered when pressing the **e** key on a selected menu entry in the boot loader menu. The **Esc** key discards any changes and reloads the standard menu interface. The **c** key loads the command line interface.

The command line interface is the most basic GRUB interface, but it is also the one that grants the most control. The command line makes it possible to type any relevant GRUB commands followed by the **Enter** key to execute them. This interface features some advanced features similar to **shell**, including **Tab** key completion based on context, and **Ctrl+a** to move to the beginning of a line and **Ctrl+e** to move to the end of a line. In addition, the **arrow**, **Home**, **End**, and **Delete** keys work as they do in the bash shell.

22.6.1. Booting to Rescue Mode

Rescue mode provides a convenient single-user environment and allows you to repair your system in situations when it is unable to complete a regular booting process. In rescue mode, the system attempts to mount all local file systems and start some important system services, but it does not activate network interfaces or allow more users to be logged into the system at the same time. In Red Hat Enterprise Linux 7, rescue mode is equivalent to single user mode and requires the root password.

1. To enter rescue mode during boot, on the GRUB 2 boot screen, press the **e** key for edit.
2. Add the following parameter at the end of the **linux** line, or **linuxefi** on UEFI systems:

```
systemd.unit=rescue.target
```

Note that equivalent parameters, **s** and **single**, can be passed to the kernel as well.

3. Press **Ctrl+x** to boot the system with the parameter.

22.6.2. Booting to Emergency Mode

Emergency mode provides the most minimal environment possible and allows you to repair your system even in situations when the system is unable to enter rescue mode. In emergency mode, the system mounts the root file system only for reading, does not attempt to mount any other local file systems, does not activate network interfaces, and only starts few essential services. In Red Hat Enterprise Linux 7, emergency mode requires the root password.

1. To enter emergency mode, on the GRUB 2 boot screen, press the **e** key for edit.
2. Add the following parameter at the end of the **linux** line, or **linuxefi** on UEFI systems:

```
systemd.unit=emergency.target
```

Note that equivalent parameters, **emergency** and **-b**, can be passed to the kernel as well.

3. Press **Ctrl+x** to boot the system with the parameter.

22.6.3. Recovering Root Password

Setting up the **root** password is a mandatory part of the Red Hat Enterprise Linux 7 installation. If you forget or lose your password, it is possible to reset it.

Note that in GRUB 2, resetting the password is no longer performed in single-user mode as it was in GRUB included in Red Hat Enterprise Linux 6. The **root** password is now required to operate in **single-user** mode as well as in **emergency** mode.

Procedure 22.1. Resetting the Root Password

1. Start the system and, on the GRUB 2 boot screen, press the **e** key for edit.
2. Add the following parameter at the end of the **linux** line, or **linuxefi** on UEFI systems:

```
init=/bin/sh
```

The Linux *kernel* will run the **/bin/sh** shell rather than the system **init** daemon. Therefore, some functions may be limited or missing.



Important

The **rhgb** and **quiet** parameters must be disabled in order to enable system messages.

3. Press **Ctrl+x** to boot the system with the parameter.

The shell prompt appears.

4. The file system is mounted read-only. You will not be allowed to change the password if the file system is not writable.

To remount the file system as writable, run the **mount -o remount, rw /** command.

5. Run the **passwd** command and follow the instructions displayed on the command line to change the **root** password.

Note that if the system is not writable, the **passwd** tool fails with the following error:

```
Authentication token manipulation error
```

6. To make sure that SELinux context of the files that were modified is restored properly after boot, run

```
touch /.autorelabel
```

7. Run the **exec /sbin/init** command to resume the initialization and finish the system boot.

Running the **exec** command with another command specified replaces the shell and creates a new process; **init** in this case.

Alternatively, if you wish to reboot the system, run the **exec /sbin/reboot** command instead.

22.7. UEFI Secure Boot

The Secure Boot technology ensures that the system firmware checks whether the system boot loader is signed with a cryptographic key authorized by a database contained in the firmware. With signature verification in the next-stage boot loader, kernel, and, potentially, user space, it is possible to prevent the execution of unsigned code.

Secure Boot is the boot path validation component of the Unified Extensible Firmware Interface (UEFI) specification. The specification defines:

- ▶ a programming interface for cryptographically protected UEFI variables in non-volatile storage,
- ▶ how the trusted X.509 root certificates are stored in UEFI variables,
- ▶ validation of UEFI applications like boot loaders and drivers,
- ▶ procedures to revoke known-bad certificates and application hashes.

UEFI Secure Boot does not prevent the installation or removal of second-stage boot loaders, nor require explicit user confirmation of such changes. Signatures are verified during booting, not when the boot loader is installed or updated. Therefore, UEFI Secure Boot does not stop boot path manipulations. It only prevents the system from executing a modified boot path once such a modification has occurred, and simplifies their detection.

22.7.1. UEFI Secure Boot Support in Red Hat Enterprise Linux 7

Red Hat Enterprise Linux 7 includes support for the UEFI Secure Boot feature, which means that Red Hat Enterprise Linux 7 can be installed and run on systems where UEFI Secure Boot is enabled. On UEFI-based systems with the Secure Boot technology enabled, all drivers that are loaded must be signed with a valid certificate, otherwise the system will not accept them. All drivers provided by Red Hat are signed by the UEFI CA certificate.

If you want to load externally built drivers — drivers that are not provided on the Red Hat Enterprise Linux DVD — you must make sure these drivers are signed as well.

Information on signing custom drivers is available in [Section 24.7, “Signing Kernel Modules for Secure Boot”](#).

22.8. Additional Resources

Please see the following resources for more information on the GRUB 2 boot loader:

- ▶ **/usr/share/doc/grub2-tools-<version-number>** — This directory contains information about using and configuring GRUB 2. <version-number> corresponds to the version of the GRUB 2 package installed.
- ▶ **info grub2** — The GRUB 2 info page contains a tutorial, a user reference manual, a programmer reference manual, and a FAQ document about GRUB 2 and its usage.
- ▶ [Red Hat Installation Guide](#) — The Installation Guide provides basic information on GRUB 2, for example, installation, terminology, interfaces, and commands.

Chapter 23. Manually Upgrading the Kernel

The Red Hat Enterprise Linux kernel is custom-built by the Red Hat Enterprise Linux kernel team to ensure its integrity and compatibility with supported hardware. Before Red Hat releases a kernel, it must first pass a rigorous set of quality assurance tests.

Red Hat Enterprise Linux kernels are packaged in the RPM format so that they are easy to upgrade and verify using the **Yum** or **PackageKit** package managers. **PackageKit** automatically queries the Red Hat Network servers and informs you of packages with available updates, including kernel packages.

This chapter is therefore *only* useful for users who need to manually update a kernel package using the **rpm** command instead of **yum**.



Warning

Whenever possible, use either the **Yum** or **PackageKit** package manager to install a new kernel because they always *install* a new kernel instead of replacing the current one, which could potentially leave your system unable to boot.



Warning

Building a custom kernel is not supported by the Red Hat Global Services Support team, and therefore is not explored in this manual.

For more information on installing kernel packages with **Yum**, see [Section 5.1.2, “Updating Packages”](#). For information on Red Hat Network, see related documents located on [Customer Portal](#).

23.1. Overview of Kernel Packages

Red Hat Enterprise Linux contains the following kernel packages:

- ▶ *kernel* — Contains the kernel for single, multicore and multiprocessor systems.
- ▶ *kernel-debug* — Contains a kernel with numerous debugging options enabled for kernel diagnosis, at the expense of reduced performance.
- ▶ *kernel-devel* — Contains the kernel headers and makefiles sufficient to build modules against the *kernel* package.
- ▶ *kernel-debug-devel* — Contains files required for building kernel modules to match the *kernel-debug* package.
- ▶ *kernel-headers* — Includes the C header files that specify the interface between the Linux kernel and user-space libraries and programs. The header files define structures and constants that are needed for building most standard programs.
- ▶ *linux-firmware* — Contains all of the firmware files that are required by various devices to operate.
- ▶ *perf* — This package contains the **perf** tool, which enables performance monitoring of the Linux kernel.
- ▶ *kernel-abi-whitelists* — Contains information pertaining to the Red Hat Enterprise Linux kernel ABI, including a lists of kernel symbols that are needed by external Linux kernel modules and a **yum** plug-in to aid enforcement.

- ▶ *kernel-tools* — Contains tools for manipulating the Linux kernel and supporting documentation.

23.2. Preparing to Upgrade

Before upgrading the kernel, it is recommended that you take some precautionary steps.

First, ensure that working boot media exists for the system. If the boot loader is not configured properly to boot the new kernel, you can use this media to boot into Red Hat Enterprise Linux.

USB media often comes in the form of flash devices sometimes called *pen drives*, *thumb disks*, or *keys*, or as an externally-connected hard disk device. Almost all media of this type is formatted as a **VFAT** file system. You can create bootable USB media on media formatted as **ext2**, **ext3**, or **VFAT**.

You can transfer a distribution image file or a minimal boot media image file to USB media. Make sure that sufficient free space is available on the device. Around **4 GB** is required for a distribution DVD image, around **700 MB** for a distribution CD image, or around **10 MB** for a minimal boot media image.

You must have a copy of the **boot.iso** file from a Red Hat Enterprise Linux installation DVD, or installation CD-ROM #1, and you need a USB storage device formatted with the **VFAT** file system and around **16 MB** of free space. The following procedure will not affect existing files on the USB storage device unless they have the same path names as the files that you copy onto it. To create USB boot media, perform the following commands as the root user:

1. Install the **syslinux** package if it is not installed on your system. To do so, as root, run the **yum install syslinux** command.
2. Install the **SYSLINUX** bootloader on the USB storage device:

```
~]# syslinux /dev/sdX1
```

...where *sdX* is the device name.

3. Create mount points for **boot.iso** and the USB storage device:

```
~]# mkdir /mnt/isoboot /mnt/diskboot
```

4. Mount **boot.iso**:

```
~]# mount -o loop boot.iso /mnt/isoboot
```

5. Mount the USB storage device:

```
~]# mount /dev/<sdX1> /mnt/diskboot
```

6. Copy the **ISOLINUX** files from the **boot.iso** to the USB storage device:

```
~]# cp /mnt/isoboot/isolinux/* /mnt/diskboot
```

7. Use the **isolinux.cfg** file from **boot.iso** as the **syslinux.cfg** file for the USB device:

```
~]# grep -v local /mnt/isoboot/isolinux/isolinux.cfg >
/mnt/diskboot/syslinux.cfg
```

8. Unmount **boot.iso** and the USB storage device:

```
~]# umount /mnt/isoboot /mnt/diskboot
```

9. You should reboot the machine with the boot media and verify that you are able to boot with it before continuing.

Alternatively, on systems with a floppy drive, you can create a boot diskette by installing the `mkbootdisk` package and running the `mkbootdisk` command as root. See `man mkbootdisk` man page after installing the package for usage information.

To determine which kernel packages are installed, execute the command `yum list installed "kernel-*"` at a shell prompt. The output will comprise some or all of the following packages, depending on the system's architecture, and the version numbers may differ:

```
~]# yum list installed "kernel-*"
kernel.x86_64          3.10.0-54.0.1.el7      @rhel7/7.0
kernel-devel.x86_64      3.10.0-54.0.1.el7     @rhel7
kernel-headers.x86_64    3.10.0-54.0.1.el7    @rhel7/7.0
```

From the output, determine which packages need to be downloaded for the kernel upgrade. For a single processor system, the only required package is the `kernel` package. See [Section 23.1, “Overview of Kernel Packages”](#) for descriptions of the different packages.

23.3. Downloading the Upgraded Kernel

There are several ways to determine if an updated kernel is available for the system.

- ▶ Security Errata — See <https://access.redhat.com/site/security/updates/active/> for information on security errata, including kernel upgrades that fix security issues.
- ▶ Via Red Hat Network — Download and install the kernel RPM packages. Red Hat Network can download the latest kernel, upgrade the kernel on the system, create an initial RAM disk image if needed, and configure the boot loader to boot the new kernel. For more information, see the related documents located at <https://access.redhat.com/site/documentation/en-US/>.

If Red Hat Network was used to download and install the updated kernel, follow the instructions in [Section 23.5, “Verifying the Initial RAM Disk Image”](#) and [Section 23.6, “Verifying the Boot Loader”](#), only do not change the kernel to boot by default. Red Hat Network automatically changes the default kernel to the latest version. To install the kernel manually, continue to [Section 23.4, “Performing the Upgrade”](#).

23.4. Performing the Upgrade

After retrieving all of the necessary packages, it is time to upgrade the existing kernel.



Important

It is strongly recommended that you keep the old kernel in case there are problems with the new kernel.

At a shell prompt, change to the directory that contains the kernel RPM packages. Use `-i` argument with the `rpm` command to keep the old kernel. Do not use the `-U` option, since it overwrites the currently installed kernel, which creates boot loader problems. For example:

```
~]# rpm -ivh kernel-<kernel_version>.<arch>.rpm
```

The next step is to verify that the initial RAM disk image has been created. See [Section 23.5, “Verifying the Initial RAM Disk Image”](#) for details.

23.5. Verifying the Initial RAM Disk Image

The job of the initial RAM disk image is to preload the block device modules, such as for IDE, SCSI or RAID, so that the root file system, on which those modules normally reside, can then be accessed and mounted. On Red Hat Enterprise Linux 7 systems, whenever a new kernel is installed using either the **Yum**, **PackageKit**, or **RPM** package manager, the **Dracut** utility is always called by the installation scripts to create an **initramfs** (initial RAM disk image).

On all architectures other than IBM eServer System i (see [Section 23.5, “Verifying the Initial RAM Disk Image and Kernel on IBM eServer System i”](#)), you can create an **initramfs** by running the **dracut** command. However, you usually don't need to create an **initramfs** manually: this step is automatically performed if the kernel and its associated packages are installed or upgraded from RPM packages distributed by Red Hat.

You can verify that an **initramfs** corresponding to your current kernel version exists and is specified correctly in the **grub.conf** configuration file by following this procedure:

Procedure 23.1. Verifying the Initial RAM Disk Image

- As root, list the contents in the **/boot/** directory and find the kernel (**vmlinuz-<kernel_version>**) and **initramfs-<kernel_version>** with the latest (most recent) version number:

Example 23.1. Ensuring that the kernel and initramfs versions match

```
~]# ls /boot/
config-3.10.0-67.el7.x86_64
config-3.10.0-78.el7.x86_64
efi
grub
grub2
initramfs-0-rescue-07f43f20a54c4ce8ada8b70d33fd001c.img
initramfs-3.10.0-67.el7.x86_64.img
initramfs-3.10.0-67.el7.x86_64kdump.img
initramfs-3.10.0-78.el7.x86_64.img
initramfs-3.10.0-78.el7.x86_64kdump.img
initrd-plymouth.img
symvers-3.10.0-67.el7.x86_64.gz
symvers-3.10.0-78.el7.x86_64.gz
System.map-3.10.0-67.el7.x86_64
System.map-3.10.0-78.el7.x86_64
vmlinuz-0-rescue-07f43f20a54c4ce8ada8b70d33fd001c
vmlinuz-3.10.0-67.el7.x86_64
vmlinuz-3.10.0-78.el7.x86_64
```

[Example 23.1, “Ensuring that the kernel and initramfs versions match”](#) shows that:

- we have three kernels installed (or, more correctly, three kernel files are present in **/boot/**),
- the latest kernel is **vmlinuz-3.10.0-78.el7.x86_64**, and

- » an **initramfs** file matching our kernel version, **initramfs-3.10.0-78.el7.x86_64kdump.img**, also exists.



Important

In the **/boot/** directory you may find several **initramfs-<version>kdump.img** files. These are special files created by the **Kdump** mechanism for kernel debugging purposes, are not used to boot the system, and can safely be ignored.

2. (Optional) If your **initramfs-<kernel_version>** file does not match the version of the latest kernel in **/boot/**, or, in certain other situations, you may need to generate an **initramfs** file with the **Dracut** utility. Simply invoking **dracut** as root without options causes it to generate an **initramfs** file in the **/boot/** directory for the latest kernel present in that directory:

```
~]# dracut
```

You must use the **--force** option if you want **dracut** to overwrite an existing **initramfs** (for example, if your **initramfs** has become corrupt). Otherwise **dracut** will refuse to overwrite the existing **initramfs** file:

```
~]# dracut
Will not override existing initramfs (/boot/initramfs-3.10.0-78.el7.x86_64.img) without --force
```

You can create an initramfs in the current directory by calling **dracut <initramfs_name> <kernel_version>**:

```
~]# dracut "initramfs-$(uname -r).img" $(uname -r)
```

If you need to specify specific kernel modules to be preloaded, add the names of those modules (minus any file name suffixes such as **.ko**) inside the parentheses of the **add_dracutmodules+="<module> [<more_modules>]"** directive of the **/etc/dracut.conf** configuration file. You can list the file contents of an **initramfs** image file created by dracut by using the **lsinitrd <initramfs_file>** command:

```
~]# lsinitrd /boot/initramfs-3.10.0-78.el7.x86_64.img
Image: /boot/initramfs-3.10.0-78.el7.x86_64.img: 11M
=====
dracut-033-68.el7
=====

drwxr-xr-x 12 root      root          0 Feb  5 06:35 .
drwxr-xr-x  2 root      root          0 Feb  5 06:35 proc
lrwxrwxrwx  1 root      root         24 Feb  5 06:35 init ->
/usr/lib/systemd/systemd
drwxr-xr-x 10 root      root          0 Feb  5 06:35 etc
drwxr-xr-x  2 root      root          0 Feb  5 06:35 usr/lib/modprobe.d
[output truncated]
```

See **man dracut** and **man dracut.conf** for more information on options and usage.

3. Examine the **grub.cfg** configuration file in the **/boot/grub2/** directory to ensure that an **initramfs initramfs-<kernel_version>.img** exists for the kernel version you are booting. See [Section 23.6, “Verifying the Boot Loader”](#) for more information.

Verifying the Initial RAM Disk Image and Kernel on IBM eServer System i

On IBM eServer System i machines, the initial RAM disk and kernel files are combined into a single file, which is created with the **addRamDisk** command. This step is performed automatically if the kernel and its associated packages are installed or upgraded from the RPM packages distributed by Red Hat; thus, it does not need to be executed manually. To verify that it was created, use the command **ls -l /boot/** to make sure the **/boot/vmlinutrd-<kernel_version>** file already exists (the **<kernel_version>** should match the version of the kernel just installed).

23.6. Verifying the Boot Loader

When you install a kernel using **rpm**, the kernel package creates an entry in the boot loader configuration file for that new kernel. However, **rpm** does *not* configure the new kernel to boot as the default kernel. You must do this manually when installing a new kernel with **rpm**.

It is always recommended to double-check the boot loader configuration file after installing a new kernel with **rpm** to ensure that the configuration is correct. Otherwise, the system may not be able to boot into Red Hat Enterprise Linux properly. If this happens, boot the system with the boot media created earlier and re-configure the boot loader.

Chapter 24. Working with Kernel Modules

The Linux kernel is modular, which means it can extend its capabilities through the use of dynamically-loaded *kernel modules*. A kernel module can provide:

- ▶ a device driver which adds support for new hardware; or,
- ▶ support for a file system such as **btrfs** or **NFS**.

Like the kernel itself, modules can take parameters that customize their behavior, though the default parameters work well in most cases. User-space tools can list the modules currently loaded into a running kernel; query all available modules for available parameters and module-specific information; and load or unload (remove) modules dynamically into or from a running kernel. Many of these utilities, which are provided by the **kmod** package, take module dependencies into account when performing operations so that manual dependency-tracking is rarely necessary.

On modern systems, kernel modules are automatically loaded by various mechanisms when the conditions call for it. However, there are occasions when it is necessary to load or unload modules manually, such as when one module is preferred over another although either could provide basic functionality, or when a module is misbehaving.

This chapter explains how to:

- ▶ use the user-space **kmod** utilities to display, query, load and unload kernel modules and their dependencies;
- ▶ set module parameters both dynamically on the command line and permanently so that you can customize the behavior of your kernel modules; and,
- ▶ load modules at boot time.



Note

In order to use the kernel module utilities described in this chapter, first ensure the **kmod** package is installed on your system by running, as root:

```
~]# yum install kmod
```

For more information on installing packages with Yum, see [Section 5.2.4, “Installing Packages”](#).

24.1. Listing Currently-Loaded Modules

You can list all kernel modules that are currently loaded into the kernel by running the **lsmod** command:

```
~]$ lsmod
Module           Size  Used by
tcp_lp          12663  0
bnep            19704  2
bluetooth      372662  7 bnep
rfkill          26536  3 bluetooth
fuse             87661  3
ip6t_rpfilter   12546  1
ip6t_REJECT     12939  2
ipt_REJECT      12541  2
xt_conntrack    12760  7
```

```

ebtable_nat          12807  0
ebtable_broute      12731  0
bridge              110196  1 ebtable_broute
stp                 12976  1 bridge
llc                14552  2 stp,bridge
ebtable_filter       12827  0
ebtables            30913  3 ebtable_broute,ebtable_nat,ebtable_filter
ip6table_nat        13015  1
nf_conntrack_ipv6   18738  5
nf_defrag_ipv6      34651  1 nf_conntrack_ipv6
nf_nat_ipv6         13279  1 ip6table_nat
ip6table_mangle     12700  1
ip6table_security   12710  1
ip6table_raw         12683  1
ip6table_filter      12815  1
ip6_tables           27025  5
ip6table_filter,ip6table_mangle,ip6table_security,ip6table_nat,ip6table_raw
iptable_nat          13011  1
nf_conntrack_ipv4   14862  4
nf_defrag_ipv4      12729  1 nf_conntrack_ipv4
nf_nat_ipv4          13263  1 iptable_nat
nf_nat               21798  4 nf_nat_ipv4,nf_nat_ipv6,ip6table_nat,iptable_nat
[output truncated]

```

Each row of **lsmod** output specifies:

- ▶ the name of a kernel module currently loaded in memory;
- ▶ the amount of memory it uses; and,
- ▶ the sum total of processes that are using the module and other modules which depend on it, followed by a list of the names of those modules, if there are any. Using this list, you can first unload all the modules depending the module you want to unload. For more information, see [Section 24.4, “Unloading a Module”](#).

Finally, note that **lsmod** output is less verbose and considerably easier to read than the content of the **/proc/modules** pseudo-file.

24.2. Displaying Information About a Module

You can display detailed information about a kernel module by running the **modinfo <module_name>** command.



Note

When entering the name of a kernel module as an argument to one of the **kmod** utilities, do not append a **.ko** extension to the end of the name. Kernel module names do not have extensions; their corresponding files do.

For example, to display information about the **e1000e** module, which is the Intel PRO/1000 network driver, run:

Example 24.1. Listing information about a kernel module with lsmod

```
~]# modinfo e1000e
filename:      /lib/modules/3.10.0-
121.el7.x86_64/kernel/drivers/net/ethernet/intel/e1000e/e1000e.ko
version:       2.3.2-k
license:        GPL
description:   Intel(R) PRO/1000 Network Driver
author:        Intel Corporation, <linux.nics@intel.com>
srcversion:    E9F7E754F6F3A1AD906634C
alias:         pci:v00008086d000015A3sv*sd*bc*sc*i*
alias:         pci:v00008086d000015A2sv*sd*bc*sc*i*
[some alias lines omitted]
alias:         pci:v00008086d0000105Esv*sd*bc*sc*i*
depends:      ptp
intree:       Y
vermagic:     3.10.0-121.el7.x86_64 SMP mod_unload modversions
signer:        Red Hat Enterprise Linux kernel signing key
sig_key:      42:49:68:9E:EF:C7:7E:95:88:0B:13:DF:E4:67:EB:1B:7A:91:D1:08
sig_hashalgo: sha256
parm:          debug:Debug level (0=none,...,16=all) (int)
parm:          copybreak:Maximum size of packet that is copied to a new buffer
on receive (uint)
parm:          TxIntDelay:Transmit Interrupt Delay (array of int)
parm:          TxAbsIntDelay:Transmit Absolute Interrupt Delay (array of int)
parm:          RxIntDelay:Receive Interrupt Delay (array of int)
parm:          RxAbsIntDelay:Receive Absolute Interrupt Delay (array of int)
parm:          InterruptThrottleRate:Interrupt Throttling Rate (array of int)
parm:          IntMode:Interrupt Mode (array of int)
parm:          SmartPowerDownEnable:Enable PHY smart power down (array of int)
parm:          KumeranLockLoss:Enable Kumeran lock loss workaround (array of int)
parm:          WriteProtectNVM:Write-protect NVM [WARNING: disabling this can
lead to corrupted NVM] (array of int)
parm:          CrcStriping:Enable CRC Stripping, disable if your BMC needs the
CRC (array of int)
```

Here are descriptions of a few of the fields in **modinfo** output:

filename

The absolute path to the **.ko** kernel object file. You can use **modinfo -n** as a shortcut command for printing only the **filename** field.

description

A short description of the module. You can use **modinfo -d** as a shortcut command for printing only the **description** field.

alias

The **alias** field appears as many times as there are aliases for a module, or is omitted entirely if there are none.

depends

This field contains a comma-separated list of all the modules this module depends on.

**Note**

If a module has no dependencies, the **depends** field may be omitted from the output.

parm

Each **parm** field presents one module parameter in the form ***parameter_name:description***, where:

- ▶ *parameter_name* is the exact syntax you should use when using it as a module parameter on the command line, or in an option line in a **.conf** file in the **/etc/modprobe.d/** directory; and,
- ▶ *description* is a brief explanation of what the parameter does, along with an expectation for the type of value the parameter accepts (such as int, unit or array of int) in parentheses.

You can list all parameters that the module supports by using the **-p** option. However, because useful value type information is omitted from **modinfo -p** output, it is more useful to run:

Example 24.2. Listing module parameters

```
~]# modinfo e1000e | grep "parm" | sort
parm:          copybreak:Maximum size of packet that is copied to a new
buffer on receive (uint)
parm:          CrcStripping:Enable CRC Stripping, disable if your BMC
needs the CRC (array of int)
parm:          debug:Debug level (0=none,...,16=all) (int)
parm:          InterruptThrottleRate:Interrupt Throttling Rate (array of
int)
parm:          IntMode:Interrupt Mode (array of int)
parm:          KumeranLockLoss:Enable Kumeran lock loss workaround (array
of int)
parm:          RxAbsIntDelay:Receive Absolute Interrupt Delay (array of
int)
parm:          RxIntDelay:Receive Interrupt Delay (array of int)
parm:          SmartPowerDownEnable:Enable PHY smart power down (array of
int)
parm:          TxAbsIntDelay:Transmit Absolute Interrupt Delay (array of
int)
parm:          TxIntDelay:Transmit Interrupt Delay (array of int)
parm:          WriteProtectNVM:Write-protect NVM [WARNING: disabling this
can lead to corrupted NVM] (array of int)
```

24.3. Loading a Module

To load a kernel module, run **modprobe <module_name>** as root. For example, to load the **wacom** module, run:

```
~]# modprobe wacom
```

By default, **modprobe** attempts to load the module from **/lib/modules/<kernel_version>/kernel/drivers/**. In this directory, each type of module has its own subdirectory, such as **net/** and **scsi/**, for network and SCSI interface drivers respectively.

Some modules have dependencies, which are other kernel modules that must be loaded before the module in question can be loaded. The **modprobe** command always takes dependencies into account when performing operations. When you ask **modprobe** to load a specific kernel module, it first examines the dependencies of that module, if there are any, and loads them if they are not already loaded into the kernel. **modprobe** resolves dependencies recursively: it will load all dependencies of dependencies, and so on, if necessary, thus ensuring that all dependencies are always met.

You can use the **-v** (or **--verbose**) option to cause **modprobe** to display detailed information about what it is doing, which can include loading module dependencies. The following is an example of loading the **Fibre Channel over Ethernet** module verbosely:

Example 24.3. modprobe -v shows module dependencies as they are loaded

```
~]# modprobe -v fcoe
insmod /lib/modules/3.10.0-121.el7.x86_64/kernel/drivers/scsi/scsi_tgt.ko
insmod /lib/modules/3.10.0-
121.el7.x86_64/kernel/drivers/scsi/scsi_transport_fc.ko
insmod /lib/modules/3.10.0-121.el7.x86_64/kernel/drivers/scsi/libfc/libfc.ko
insmod /lib/modules/3.10.0-121.el7.x86_64/kernel/drivers/scsi/fcoe/libfcoe.ko
insmod /lib/modules/3.10.0-121.el7.x86_64/kernel/drivers/scsi/fcoe.ko
```

[Example 24.3, “modprobe -v shows module dependencies as they are loaded”](#) shows that **modprobe** loaded the **scsi_tgt**, **scsi_transport_fc**, **libfc**, and **libfcoe** modules as dependencies before finally loading **fcoe**. Also note that **modprobe** used the more primitive **insmod** command to insert the modules into the running kernel.



Important

Although the **insmod** command can also be used to load kernel modules, it does not resolve dependencies. Because of this, you should *always* load modules using **modprobe** instead.

24.4. Unloading a Module

You can unload a kernel module by running **modprobe -r <module_name>** as root. For example, assuming that the **wacom** module is already loaded into the kernel, you can unload it by running:

```
~]# modprobe -r wacom
```

However, this command will fail if a process is using:

- ▶ the **wacom** module;
- ▶ a module that **wacom** directly depends on, or;
- ▶ any module that **wacom**, through the dependency tree, depends on indirectly.

See [Section 24.1, “Listing Currently-Loaded Modules”](#) for more information about using **lsmod** to obtain the names of the modules which are preventing you from unloading a certain module.

For example, if you want to unload the **firewire_ohci** module, your terminal session might look similar to this:

```
~]# modinfo -F depends firewire_ohci
firewire-core
~]# modinfo -F depends firewire_core
crc-itu-t
~]# modinfo -F depends crc-itu-t
```

You have figured out the dependency tree (which does not branch in this example) for the loaded Firewire modules: **firewire_ohci** depends on **firewire_core**, which itself depends on **crc-itu-t**.

You can unload **firewire_ohci** using the **modprobe -v -r <module_name>** command, where **-r** is short for **--remove** and **-v** for **--verbose**:

```
~]# modprobe -r -v firewire_ohci
rmmod firewire_ohci
rmmod firewire_core
rmmod crc_itu_t
```

The output shows that modules are unloaded in the reverse order that they are loaded, given that no processes depend on any of the modules being unloaded.



Important

Although the **rmmod** command can be used to unload kernel modules, it is recommended to use **modprobe -r** instead.

24.5. Setting Module Parameters

Like the kernel itself, modules can also take parameters that change their behavior. Most of the time, the default ones work well, but occasionally it is necessary or desirable to set custom parameters for a module. Because parameters cannot be dynamically set for a module that is already loaded into a running kernel, there are two different methods for setting them.

1. You can unload all dependencies of the module you want to set parameters for, unload the module using **modprobe -r**, and then load it with **modprobe** along with a list of customized parameters. This method is often used when the module does not have many dependencies, or to test different combinations of parameters without making them persistent, and is the method covered in this section.
2. Alternatively, you can list the new parameters in an existing or newly created file in the **/etc/modprobe.d/** directory. This method makes the module parameters persistent by ensuring that they are set each time the module is loaded, such as after every reboot or **modprobe** command. This method is covered in [Section 24.6, “Persistent Module Loading”](#), though the following information is a prerequisite.

You can use **modprobe** to load a kernel module with custom parameters using the following command line format:

Example 24.4. Supplying optional parameters when loading a kernel module

```
~]# modprobe <module_name> [parameter=value]
```

When loading a module with custom parameters on the command line, be aware of the following:

- ▶ You can enter multiple parameters and values by separating them with spaces.
- ▶ Some module parameters expect a list of comma-separated values as their argument. When entering the list of values, do *not* insert a space after each comma, or **modprobe** will incorrectly interpret the values following spaces as additional parameters.
- ▶ The **modprobe** command silently succeeds with an exit status of **0** if:
 - it successfully loads the module, or
 - the module is *already* loaded into the kernel.

Thus, you must ensure that the module is not already loaded before attempting to load it with custom parameters. The **modprobe** command does not automatically reload the module, or alert you that it is already loaded.

Here are the recommended steps for setting custom parameters and then loading a kernel module. This procedure illustrates the steps using the **e1000e** module, which is the network driver for Intel PRO/1000 network adapters, as an example:

Procedure 24.1. Loading a Kernel Module with Custom Parameters

1.

First, ensure the module is not already loaded into the kernel:

```
~]# lsmod |grep e1000e
~]#
```

Output would indicate that the module is already loaded into the kernel, in which case you must first unload it before proceeding. See [Section 24.4. “Unloading a Module”](#) for instructions on safely unloading it.

2.

Load the module and list all custom parameters after the module name. For example, if you wanted to load the Intel PRO/1000 network driver with the interrupt throttle rate set to 3000 interrupts per second for the first, second, and third instances of the driver, and turn on debug, you would run, as **root**:

```
~]# modprobe e1000e InterruptThrottleRate=3000,3000,3000 debug=1
```

This example illustrates passing multiple values to a single parameter by separating them with commas and omitting any spaces between them.

24.6. Persistent Module Loading

As shown in [Example 24.1, “Listing information about a kernel module with lsmod”](#), many kernel modules are loaded automatically at boot time. You can specify additional modules to be loaded by creating a new **<file_name>.modules** file in the **/etc/sysconfig/modules/** directory, where **<file_name>** is any descriptive name of your choice. Your **<file_name>.modules** files are treated by the system startup scripts as shell scripts, and as such should begin with an *interpreter directive* (also called a “bang line”) as their first line:

Example 24.5. First line of a `file_name.modules` file

```
#!/bin/sh
```

Additionally, the `<file_name>.modules` file should be executable. You can make it executable by running:

```
modules]# chmod +x <file_name>.modules
```

For example, the following `bluez-uinput.modules` script loads the `uinput` module:

Example 24.6. `/etc/sysconfig/modules/bluez-uinput.modules`

```
#!/bin/sh

if [ ! -c /dev/input/uinput ] ; then
    exec /sbin/modprobe uinput >/dev/null 2>&1
fi
```

The `if`-conditional statement on the third line ensures that the `/dev/input/uinput` file does *not* already exist (the `!` symbol negates the condition), and, if that is the case, loads the `uinput` module by calling `exec /sbin/modprobe uinput`. Note that the `uinput` module creates the `/dev/input/uinput` file, so testing to see if that file exists serves as verification of whether the `uinput` module is loaded into the kernel.

The following `>/dev/null 2>&1` clause at the end of that line redirects any output to `/dev/null` so that the `modprobe` command remains quiet.

24.7. Signing Kernel Modules for Secure Boot

Red Hat Enterprise Linux 7 includes support for the UEFI Secure Boot feature, which means that Red Hat Enterprise Linux 7 can be installed and run on systems where UEFI Secure Boot is enabled. [3] When Secure Boot is enabled, the EFI operating system boot loaders, the Red Hat Enterprise Linux kernel, and all kernel modules must be signed with a private key and authenticated with the corresponding public key. The Red Hat Enterprise Linux 7 distribution includes signed boot loaders, signed kernels, and signed kernel modules. In addition, the signed first-stage boot loader and the signed kernel include embedded Red Hat public keys. These signed executable binaries and embedded keys enable Red Hat Enterprise Linux 7 to install, boot, and run with the Microsoft UEFI Secure Boot CA keys that are provided by the UEFI firmware on systems that support UEFI Secure Boot. [4]

The information provided in the following sections describes steps necessary to enable you to self-sign privately built kernel modules for use with Red Hat Enterprise Linux 7 on UEFI-based systems where Secure Boot is enabled. These sections also provide an overview of available options for getting your public key onto the target system where you wish to deploy your kernel module.

24.7.1. Prerequisites

In order to enable signing of externally built modules, the tools listed in the following table are required to be installed on the system.

Table 24.1. Required Tools

Tool	Provided by Package	Used on	Purpose
openssl	<i>openssl</i>	Build system	Generates public and private X.509 key pair
sign-file	<i>kernel-devel</i>	Build system	Perl script used to sign kernel modules
perl	<i>perl</i>	Build system	Perl interpreter used to run the signing script
mokutil	<i>mokutil</i>	Target system	Optional tool used to manually enroll the public key
keyctl	<i>keyutils</i>	Target system	Optional tool used to display public keys in the system key ring



Note

Note that the build system, where you build and sign your kernel module, does not need to have UEFI Secure Boot enabled and does not even need to be a UEFI-based system.

24.7.2. Kernel Module Authentication

In Red Hat Enterprise Linux 7, when a kernel module is loaded, the module's signature is checked using the public x.509 keys on the kernel's system key ring, excluding those keys that are on the kernel's system black list key ring.

24.7.2.1. Sources For Public Keys Used To Authenticate Kernel Modules

During boot, the kernel loads X.509 keys into the system key ring or the system black list key ring from a set of persistent key stores as shown in [Table 24.2, “Sources For System Key Rings”](#).

Table 24.2. Sources For System Key Rings

Source of X.509 Keys	User Ability to Add Keys	UEFI Secure Boot State	Keys Loaded During Boot
Embedded in kernel	No	-	.system_keyring
UEFI Secure Boot "db"	Limited	Not enabled	No
		Enabled	.system_keyring
UEFI Secure Boot "dbx"	Limited	Not enabled	No
		Enabled	.system_keyring
Embedded in shim.EFI boot loader	No	Not enabled	No
		Enabled	.system_keyring
Machine Owner Key (MOK) list	Yes	Not enabled	No
		Enabled	.system_keyring

Note that if the system is not UEFI-based or if UEFI Secure Boot is not enabled, then only the keys that are embedded in the kernel are loaded onto the system key ring and you have no ability to augment that set of keys without rebuilding the kernel. The system black list key ring is a list of X.509 keys which have been revoked. If your module is signed by a key on the black list then it will fail authentication even if your public key is in the system key ring.

You can display information about the keys on the system key rings using the **keyctl** utility. The following is abbreviated example output from a Red Hat Enterprise Linux 7 system where UEFI Secure Boot is not enabled.

```
~]# keyctl list %:.system_keyring
3 keys in keyring:
...asymmetric: Red Hat Enterprise Linux Driver Update Program (key 3): bf57f3e87...
...asymmetric: Red Hat Enterprise Linux kernel signing key: 4249689eefc77e95880b...
...asymmetric: Red Hat Enterprise Linux kpatch signing key: 4d38fd864ebe18c5f0b7...
```

The following is abbreviated example output from a Red Hat Enterprise Linux 7 system where UEFI Secure Boot is enabled.

```
~]# keyctl list %:.system_keyring
6 keys in keyring:
...asymmetric: Red Hat Enterprise Linux Driver Update Program (key 3): bf57f3e87...
...asymmetric: Red Hat Secure Boot (CA key 1): 4016841644ce3a810408050766e8f8a29...
...asymmetric: Microsoft Corporation UEFI CA 2011: 13adbf4309bd82709c8cd54f316ed...
...asymmetric: Microsoft Windows Production PCA 2011: a92902398e16c49778cd90f99e...
...asymmetric: Red Hat Enterprise Linux kernel signing key: 4249689eefc77e95880b...
...asymmetric: Red Hat Enterprise Linux kpatch signing key: 4d38fd864ebe18c5f0b7...
```

The above output shows the addition of two keys from the UEFI Secure Boot "db" keys plus the "Red Hat Secure Boot(CA key 1)" which is embedded in the **shim.efi** boot loader. You can also look for the kernel console messages that identify the keys with an UEFI Secure Boot related source, that is UEFI Secure Boot db, embedded shim, and MOK list.

```
~]# dmesg | grep 'EFI: Loaded cert'
[5.160660] EFI: Loaded cert 'Microsoft Windows Production PCA 2011: a9290239...
[5.160674] EFI: Loaded cert 'Microsoft Corporation UEFI CA 2011: 13adbf4309b...
[5.165794] EFI: Loaded cert 'Red Hat Secure Boot (CA key 1): 4016841644ce3a8...
```

24.7.2.2. Kernel Module Authentication Requirements

If UEFI Secure Boot is enabled or if the **module.sig_enforce** kernel parameter has been specified, then only signed kernel modules that are authenticated using a key on the system key ring can be successfully loaded. [5] If UEFI Secure Boot is disabled and if the **module.sig_enforce** kernel parameter has not been specified, then unsigned kernel modules and signed kernel modules without a public key can be successfully loaded. This is summarized in [Table 24.3, “Kernel Module Authentication Requirements for Loading”](#).

Table 24.3. Kernel Module Authentication Requirements for Loading

Module Signed	Public Key Found and Signature Valid	UEFI Secure Boot State	module.sig_enforce	Module Load	Kernel Tainted
Unsigned	-	Not enabled	Not enabled	Succeeds	Yes
		Not enabled	Enabled	Fails	
		Enabled	-	Fails	-
Signed	No	Not enabled	Not enabled	Succeeds	Yes
		Not enabled	Enabled	Fails	-
		Enabled	-	Fails	-
Signed	Yes	Not enabled	Not enabled	Succeeds	No

Module Signed	Public Key Found and Signature Valid	UEFI Secure Boot State	module.sig_enforce	Module Load	Kernel Tainted
		Not enabled	Enabled	Succeeds	No
		Enabled	-	Succeeds	No

Subsequent sections will describe how to generate your own public/private X.509 key pair, how to use the private key to sign your kernel module, and how to enroll the public key into a source for the system key ring.

24.7.3. Generating a Public and Private X.509 Key Pair

You need to generate a public and private X.509 key pair that will be used to sign your kernel module after you have built it. The corresponding public key will be used to authenticate your kernel module when it is loaded.

1. The **openssl** tool can be used to generate a key pair that satisfies the requirements for kernel module signing in Red Hat Enterprise Linux 7. Some of the parameters for this key generation request are best specified with a configuration file; please follow the example below to create your own configuration file.

```
~]# cat << EOF > configuration_file.config
[ req ]
default_bits = 4096
distinguished_name = req_distinguished_name
prompt = no
string_mask = utf8only
x509_extensions = myexts

[ req_distinguished_name ]
O = Organization
CN = Organization signing key
emailAddress = E-mail address

[ myexts ]
basicConstraints=critical,CA:FALSE
keyUsage=digitalSignature
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid
EOF
```

2. After you have created the configuration file, you can create an X.509 public and private key pair. The public key will be written to the **<file_name>.der** file and the private key will be written to the **<file_name>.priv** file.

```
openssl req -x509 -new -nodes -utf8 -sha256 -days 36500 \
> -batch -config configuration_file.config -outform DER \
> -out public_key.der \
> -keyout private_key.priv
```

3. Enroll your public key on all systems where you wish to authenticate and load your kernel module.



Warning

Take proper care to guard the contents of your private key. In the wrong hands, the key could be used to compromise any system which has your public key.

24.7.4. Enrolling Public Key on Target System

When Red Hat Enterprise Linux 7 boots on a UEFI-based system with Secure Boot enabled, all keys that are in the Secure Boot db key database, but not in the dbx database of revoked keys, are loaded onto the system keyring by the kernel. The system keyring is used to authenticate kernel modules.

24.7.4.1. Factory Firmware Image Including Public Key

The easiest way to facilitate authentication of your kernel module on the target system is the system vendor to incorporate your public key into the UEFI Secure Boot key database in their factory firmware image.

24.7.4.2. Executable Key Enrollment Image Adding Public Key

It is possible to add a key to an existing populated and active Secure Boot key database. This can be done by writing and providing an EFI executable *enrollment* image. Such an enrollment image contains a properly formed request to append a key to the Secure Boot key database. This request must include a key that is properly signed by the private key that corresponds to a public key that is already in the system's Secure Boot Key Exchange Key (KEK) database. Additionally, this EFI image must be signed by a private key that corresponds to a public key that is already in the key database.

It is also possible to write an enrollment image that runs under Red Hat Enterprise Linux 7. However, the Red Hat Enterprise Linux 7 image must be properly signed by a private key that corresponds to a public key that is already in the KEK database.

As such, the construction of either type of key enrollment images requires assistance from the platform vendor.

24.7.4.3. System Administrator Manually Adding Public Key to the MOK List

The Machine Owner Key (MOK) facility is a feature that is supported by Red Hat Enterprise Linux 7 and can be used to augment the UEFI Secure Boot key database. When Red Hat Enterprise Linux 7 boots on a UEFI-enabled system with Secure Boot enabled, the keys on the MOK list are also added to the system keyring in addition to the keys from the key database. The MOK list keys are also stored persistently and securely in the same fashion as the Secure Boot key database keys, but these are two separate facilities. The MOK facility is supported by `shim.efi`, `MokManager.efi`, `grubx64.efi`, and the Red Hat Enterprise Linux 7 `mokutil` utility.

The major capability provided by the MOK facility is the ability to add public keys to the MOK list without needing to have the key chain back to another key that is already in the KEK database. However, enrolling a MOK key requires manual interaction by a *physically present* user at the UEFI system console on each target system. Nevertheless, the MOK facility provides an excellent method for testing newly generated key pairs and testing kernel modules signed with them.

Follow these steps to add your public key to the MOK list:

1. Request addition of your public key to the MOK list using a Red Hat Enterprise Linux 7 userspace utility:

```
~]# mokutil --import my_signing_key_pub.der
```

You will be asked to enter and confirm a password for this MOK enrollment request.

2. Reboot the machine.
3. The pending MOK key enrollment request will be noticed by **shim.efi** and it will launch **MokManager.efi** to allow you to complete the enrollment from the UEFI console. You will need to enter the password you previously associated with this request and confirm the enrollment. Your public key is added to the MOK list, which is persistent.

Once a key is on the MOK list, it will be automatically propagated to the system key ring on this and subsequent boots when UEFI Secure Boot is enabled.

24.7.5. Signing Kernel Module with the Private Key

There are no extra steps required to prepare your kernel module for signing. You build your kernel module normally. Assuming an appropriate Makefile and corresponding sources, follow these steps to build your module and sign it:

1. Build your **my_module.ko** module the standard way:

```
~]# make -C /usr/src/kernels/$(uname -r) M=$PWD modules
```

2. Sign your kernel module with your private key. This is done with a Perl script. Note that the script requires that you provide both the files that contain your private and the public key as well as the kernel module file that you wish to sign.

```
~]# perl /usr/src/kernels/$(uname -r)/scripts/sign-file \
> sha256 \
> my_signing_key.priv \
> my_signing_key_pub.der \
> my_module.ko
```

Your kernel module is in ELF image format and this script computes and appends the signature directly to the ELF image in your **my_module.ko** file. The **modinfo** utility can be used to display information about the kernel module's signature, if it is present. For information on using the utility, see [Section 24.2, “Displaying Information About a Module”](#).

Note that this appended signature is not contained in an ELF image section and is not a formal part of the ELF image. Therefore, tools such as **readelf** will not be able to display the signature on your kernel module.

Your kernel module is now ready for loading. Note that your signed kernel module is also loadable on systems where UEFI Secure Boot is disabled or on a non-UEFI system. That means you do not need to provide both a signed and unsigned version of your kernel module.

24.7.6. Loading Signed Kernel Module

Once your public key is enrolled and is in the system keyring, the normal kernel module loading mechanisms will work transparently. In the following example, you will use **mokutil** to add your public key to the MOK list and you will manually load your kernel module with **modprobe**.

1. Optionally, you can verify that your kernel module will not load before you have enrolled your public key. First, verify what keys have been added to the system key ring on the current boot by running the **keyctl list %: .system_keyring** as root. Since your public key has not been enrolled yet, it should not be displayed in the output of the command.
2. Request enrollment of your public key.

```
~]# mokutil --import my_signing_key_pub.der
```

3. Reboot, and complete the enrollment at the UEFI console.

```
~]# reboot
```

4. After the system reboots, verify the keys on the system key ring again.

```
~]# keyctl list %:.system_keyring
```

5. You should now be able to load your kernel module successfully.

```
~]# modprobe -v my_module
insmod /lib/modules/3.10.0-123.el7.x86_64/extr/my_module.ko
~]# lsmod | grep my_module
my_module 12425 0
```

24.8. Additional Resources

For more information on kernel modules and their utilities, see the following resources.

Manual Page Documentation

- ▶ **man lsmod** — The manual page for the **lsmod** command.
- ▶ **man modinfo** — The manual page for the **modinfo** command.
- ▶ **man modprobe** — The manual page for the **modprobe** command.
- ▶ **man rmmod** — The manual page for the **rmmod** command.
- ▶ **man ethtool** — The manual page for the **ethtool** command.
- ▶ **man mii-tool** — The manual page for the **mii-tool** command.

Installable and External Documentation

- ▶ [Linux Loadable Kernel Module HOWTO](#) — The *Linux Loadable Kernel Module HOWTO* from the Linux Documentation Project contains further information on working with kernel modules.

[3] Red Hat Enterprise Linux 7 does not require the use of Secure Boot on UEFI systems.

[4] Not all UEFI-based systems include support for Secure Boot.

[5] Provided that the public key is not on the system black list key ring.

RPM

The **RPM Package Manager (RPM)** is an open packaging system that runs on Red Hat Enterprise Linux as well as other Linux and UNIX systems. Red Hat and the Fedora Project encourage other vendors to use **RPM** for their own products. **RPM** is distributed under the terms of the *GPL (GNU General Public License)*.

The **RPM Package Manager** only works with packages built in the *RPM format*. **RPM** itself is provided as the pre-installed *rpm* package. For the end user, **RPM** makes system updates easy. Installing, uninstalling, and upgrading **RPM** packages can be accomplished with short commands. **RPM** maintains a database of installed packages and their files, so you can invoke powerful queries and verifications on your system. There are several applications, such as **Yum** or **PackageKit**, that can make working with packages in the **RPM** format even easier.



Warning

For most package-management tasks, the **Yum** package manager offers equal and often greater capabilities and utility than **RPM**. **Yum** also performs and tracks complicated system-dependency resolutions. **Yum** maintains the system integrity and forces a system integrity check if packages are installed or removed using another application, such as **RPM**, instead of **Yum**. For these reasons, it is highly recommended that you use **Yum** instead of **RPM** whenever possible to perform package-management tasks. See [Chapter 5, Yum](#).

If you prefer a graphical interface, you can use the **PackageKit** GUI application, which uses **Yum** as its back end, to manage your system's packages.

During upgrades, **RPM** handles configuration files carefully, so that you never lose your customizations—something that you cannot accomplish with regular **.tar.gz** files.

For the developer, **RPM** allows you to take software source code and package it into source and binary packages for end users. This process is quite simple and is driven from a single file and optional patches that you create. This clear delineation between pristine sources and your patches along with build instructions eases the maintenance of the package as new versions of the software are released.



Note

Because **RPM** can make changes to the system itself, performing operations like installing, upgrading, downgrading, and uninstalling binary packages system-wide requires **root** privileges in most cases.

A.1. RPM Design Goals

To understand how to use **RPM**, it is helpful to understand the design goals of **RPM**:

Upgradability

With **RPM**, you can upgrade individual components of your system without a complete reinstallation. When you get a new release of an operating system based on **RPM**, such as Red Hat Enterprise Linux, you do not need to reinstall a fresh copy of the operating system on your machine (as you might need to with operating systems based on other packaging systems).

RPM allows for intelligent, fully-automated, in-place upgrades of your system. In addition, configuration files in packages are preserved across upgrades, so you do not lose your customizations. There are no special upgrade files needed to upgrade a package because the same **RPM** file is used to both install and upgrade the package on your system.

Powerful Querying

RPM is designed to provide powerful querying options. You can perform searches on your copy of the database for packages or even just certain files. You can also easily find out what package a file belongs to and where the package came from. The files an **RPM** package contains are in a compressed archive, with a custom binary header containing useful information about the package and its contents, allowing you to query individual packages quickly and easily.

System Verification

Another powerful **RPM** feature is the ability to verify packages. It allows you to verify that the files installed on your system are the same as the ones supplied by a given package. If an inconsistency is detected, **RPM** notifies you, and you can reinstall the package if necessary. Any configuration files that you modified are preserved during reinstallation.

Pristine Sources

A crucial design goal is to allow the use of *pristine* software sources, as distributed by the original authors of the software. With **RPM**, you have the pristine sources along with any patches that were used, plus complete build instructions. This is an important advantage for several reasons. For instance, if a new version of a program is released, you do not necessarily have to start from scratch to get it to compile. You can look at the patch to see what you *might* need to do. All the compiled-in defaults, and all of the changes that were made to get the software to build properly, are easily visible using this technique.

The goal of keeping sources pristine may seem important only for developers, but it results in higher quality software for end users.

A.2. Using RPM

RPM has five basic modes of operation (excluding package building): installing, uninstalling, upgrading, querying, and verifying. This section contains an overview of each mode. For complete details and options, try **rpm --help** or see **rpm(8)**. Also, see [Section A.5, “Additional Resources”](#) for more information on **RPM**.

A.2.1. Installing and Upgrading Packages

RPM packages typically have file names like **tree-1.6.0-10.el7.x86_64.rpm**. The file name includes the package name (**tree**), version (**1.6.0**), release (**10**), operating system major version (**el7**) and CPU architecture (**x86_64**).



Important

When installing a package, ensure it is compatible with your operating system and processor architecture. This can usually be determined by checking the package name. For example, the file name of an **RPM** package compiled for the AMD64/Intel 64 computer architectures ends with **x86_64.rpm**.

You can use the **-U** (or **--upgrade**) option to:

- ▶ upgrade an existing package on the system to a newer version, or
- ▶ install a package if an older version is not already installed.

That is, **rpm -U package.rpm** is able to perform the function of either *upgrading* or *installing* as is appropriate for the package.

Assuming the **tree-1.6.0-10.el7.x86_64.rpm** package is in the current directory, log in as **root** and type the following command at a shell prompt to either upgrade or install the *tree* package:

```
~]# rpm -Uvh tree-1.6.0-10.el7.x86_64.rpm
```

Note

The **-v** and **-h** options (which are combined with **-U**) cause **rpm** to print a more verbose output and display a progress meter using hash signs.

If the upgrade or installation is successful, the following output is displayed:

```
Preparing...                                          ###### [100%]
Updating / installing...
 1:tree-1.6.0-10.el7                            ###### [100%]
```

Warning

rpm provides two different options for installing packages: the aforementioned **-U** option (which historically stands for *upgrade*), and the **-i** option (which historically stands for *install*). Because the **-U** option includes both install and upgrade functions, we recommend the use of **rpm -Uvh** with all packages except *kernel packages*.

You should always use the **-i** option to *install* a new kernel package instead of upgrading it. This is because using the **-U** option to upgrade a kernel package removes the previous (older) kernel package, which could render the system unable to boot if there is a problem with the new kernel. Therefore, use the **rpm -i kernel_package** command to install a new kernel *without replacing any older kernel packages*. For more information on installing *kernel* packages, see [Chapter 23, Manually Upgrading the Kernel](#).

The signature of a package is checked automatically when installing or upgrading a package. The signature confirms that the package was signed by an authorized party. If the verification of the signature fails, an error message is displayed.

If you do not have the appropriate key installed to verify the signature, the message contains the word **NOKEY**:

```
warning: tree-1.6.0-10.el7.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID
431d51: NOKEY
```

See [Section A.3.2, “Checking Package Signatures”](#) for more information on checking package signatures.

A.2.1.1. Replacing Already-Installed Packages

If a package of the same name and version is already installed, the following output is displayed:

```
Preparing... ################################################ [100%]
package tree-1.6.0-10.el7.x86_64 is already installed
```

To install the package anyway, use the **--replacepkgs** option, which tells **RPM** to ignore the error:

```
~]# rpm -Uvh --replacepkgs tree-1.6.0-10.el7.x86_64.rpm
```

This option is helpful if files installed from the package were deleted or if you want the original configuration files to be installed.

If you attempt an upgrade to an *older* version of a package (that is, if a newer version of the package is already installed), **RPM** informs you that a newer version is already installed. To force **RPM** to perform the downgrade, use the **--oldpackage** option:

```
rpm -Uvh --oldpackage older_package.rpm
```

A.2.1.2. Resolving File Conflicts

If you attempt to install a package that contains a file that has already been installed by another package, a conflict message is displayed. To make **RPM** ignore this error, use the **--replacefiles** option:

```
rpm -Uvh --replacefiles package.rpm
```

A.2.1.3. Satisfying Unresolved Dependencies

RPM packages sometimes depend on other packages, which means that they require other packages to be installed to run properly. If you try to install a package that has an unresolved dependency, a message about a failed dependency is displayed.

Find the suggested package(s) on the Red Hat Enterprise Linux installation media or on one of the active Red Hat Enterprise Linux mirrors and add it to the installation command. To determine which package contains the required file, use the **--whatprovides** option:

```
rpm -q --whatprovides "required_file"
```

If the package that contains *required_file* is in the **RPM** database, the name of the package is displayed:



Warning

Although you can *force* **rpm** to install a package that has an unresolved dependency (using the **--nodeps** option), this is *not* recommended and will usually result in the installed software failing to run. Installing packages with **--nodeps** can cause applications to misbehave or crash. It can also cause serious package-management problems or system failure. For these reasons, it is best to heed the warnings about missing dependencies. The **Yum** package manager performs automatic dependency resolution and fetches dependencies from online repositories.

A.2.1.4. Preserving Changes in Configuration Files

Because **RPM** performs intelligent upgrading of packages with configuration files, you may see the following message:

```
saving /etc/configuration_file.conf as /etc/configuration_file.conf.rpm.save
```

This message means that the changes you made to the configuration file may not be *forward-compatible* with the new configuration file in the package, so **RPM** saved your original file and installed a new one. You should investigate the differences between the two configuration files and resolve them as soon as possible to ensure that your system continues to function properly.

Alternatively, **RPM** may save the package's *new* configuration file as, for example, **configuration_file.conf.rpmnew** and leave the configuration file you modified untouched. You should still resolve any conflicts between your modified configuration file and the new one, usually by merging changes from the old one to the new one, for example using the **diff** program.

A.2.2. Uninstalling Packages

Uninstalling a package is just as simple as installing one. Type the following command at a shell prompt as **root**:

```
rpm -e package
```

Note

Note that the command expects only the package *name*, not the name of the original package *file*. If you attempt to uninstall a package using the **rpm -e** command and provide the original full file name, you receive a package-name error.

You can encounter dependency errors when uninstalling a package if another installed package depends on the one you are trying to remove. For example:

```
~]# rpm -e ghostscript
error: Failed dependencies:
        ghostscript is needed by (installed) ghostscript-cups-9.07-16.el7.x86_64
        ghostscript is needed by (installed) foomatic-4.0.9-6.el7.x86_64
        libgs.so.9()(64bit) is needed by (installed) libspectre-0.2.7-4.el7.x86_64
        libijs-0.35.so()(64bit) is needed by (installed) gutenprint-5.2.9-
15.el7.x86_64
        libijs-0.35.so()(64bit) is needed by (installed) cups-filters-1.0.35-
15.el7.x86_64
```

Warning

Although you can *force* **rpm** to uninstall a package that has unresolved dependencies (using the **-nodeps** option), this is *not* recommended. Removing packages with **--nodeps** can cause applications from the packages whose dependencies are removed to misbehave or crash. It can also cause serious package-management problems or system failure. For these reasons, it is best to heed the warnings about failed dependencies.

A.2.3. Freshening Packages

Freshening is similar to upgrading, except that only installed packages are upgraded. Type the following command at a shell prompt as **root**:

```
rpm -Fvh package.rpm
```

The **-F** (or **--freshen**) option compares the versions of the packages specified on the command line with the versions of packages that are already installed on your system. When a newer version of an already-installed package is processed by the **--freshen** option, it is upgraded to the newer version. However, the **--freshen** option does not install a package if no previously-installed package of the same name exists. This differs from regular upgrading, as an upgrade installs all specified packages regardless of whether or not older versions of the packages are already installed.

Freshening works for single packages or package groups. For example, freshening can help if you download a large number of different packages, and you only want to upgrade those packages that are already installed on your system. In this case, issue the following command with the *** .rpm** glob:

```
~]# rpm -Fvh *.rpm
```

RPM then automatically upgrades only those packages that are already installed.

A.2.4. Querying Packages

The **RPM** database stores information about all **RPM** packages installed on your system. It is stored in the **/var/lib/rpm/** directory and is used for many things, including querying what packages are installed, what version each package is, and for calculating changes to files in packages since their installation. To query this database, use the **rpm** command with the **-q** (or **--query**) option:

```
rpm -q package_name
```

This command displays the package name, version, and release number of the installed package *package_name*. For example:

```
~]$ rpm -q tree
tree-1.6.0-10.el7.x86_64
```

See the **Package Selection Options** subheading in the **rpm(8)** manual page for a list of options that can be used to further refine or qualify your query. Use options listed below the **Package Query Options** subheading to specify what information to display about the queried packages.

A.2.5. Verifying Packages

Verifying a package compares information about files installed from a package with the same information from the original package. Among other things, verifying compares the file size, MD5 sum, permissions, type, owner, and the group of each file.

Use the **rpm** command with the **-V** (or **--verify**) option to verify packages. For example:

```
~]$ rpm -V tree
```

See the **Package Selection Options** subheading in the **rpm(8)** manual page for a list of options that can be used to further refine or qualify your query. Use options listed below the **Verify Options** subheading to specify what characteristics to verify in the queried packages.

If everything verifies properly, there is no output. If there are any discrepancies, they are displayed. The output consists of lines similar to these:

```
~]# rpm -V abrt
S.5....T.  c /etc/abrt/abrt.conf
.M.....    /var/spool/abrt-upload
```

The format of the output is a string of nine characters followed by an optional attribute marker and then the name of the processed file.

The first nine characters are the results of tests performed on the file. Each test is the comparison of one attribute of the file to the value of that attribute as recorded in the **RPM** database. A single period (.) means the test passed, and the question-mark character (?) signifies that the test could not be performed. The following table lists symbols that denote specific discrepancies:

Table A.1. RPM Verification Symbols

Symbol	Description
S	file size differs
M	mode differs (includes permissions and file type)
5	digest (formerly MD5 sum) differs
D	device major/minor number mismatch
L	readLink(2) path mismatch
U	user ownership differs
G	group ownership differs
T	mtime differs
P	capabilities differ

The attribute marker, if present, describes the purpose of the given file. The following table lists the available attribute markers:

Table A.2. RPM Verification Symbols

Marker	Description
c	configuration file
d	documentation file
l	license file
r	readme file

If you see any output, use your best judgment to determine if you should remove the package, reinstall it, or fix the problem in another way.

A.3. Finding and Verifying RPM Packages

Before using any **RPM** packages, you must know where to find them and how to know if you can trust them.

A.3.1. Finding RPM Packages

Although there are many **RPM** repositories on the Internet, for security and compatibility reasons, you should consider installing only official Red Hat-provided RPM packages. The following is a list of sources for **RPM** packages:

- ▶ Official Red Hat Enterprise Linux installation media.
- ▶ Official **RPM** repositories provided with the **Yum** package manager. See [Chapter 5, Yum](#) for details on how to use the official Red Hat Enterprise Linux package repositories.

- ▶ The Red Hat Errata Page, available on the Customer Portal at <https://rhn.redhat.com/rhn/errata/RelevantErrata.do>.
- ▶ Extra Packages for Enterprise Linux (EPEL) is a community effort to provide a repository with high-quality add-on packages for Red Hat Enterprise Linux. See <http://fedoraproject.org/wiki/EPEL> for details on EPEL **RPM** packages.
- ▶ Unofficial, third-party repositories not affiliated with Red Hat also provide RPM packages.



Important

When considering third-party repositories for use with your Red Hat Enterprise Linux system, pay close attention to the repository's web site with regard to package compatibility before adding the repository as a package source. Alternate package repositories may offer different, incompatible versions of the same software, including packages already included in the Red Hat Enterprise Linux repositories.

A.3.2. Checking Package Signatures

RPM packages can be signed using **GNU Privacy Guard** (or **GPG**), which helps you make certain that downloaded packages are trustworthy. **GPG** is a tool for secure communication. With **GPG**, you can authenticate the validity of documents and encrypt or decrypt data.

To verify that a package has not been corrupted or tampered with, check its **GPG** signature by using the **rpmkeys** command with the **-K** (or **--checksig**) option:

```
rpmkeys -K package.rpm
```

Note that the **Yum** package manager performs automatic checking of **GPG** signatures during installations and upgrades.

GPG is installed by default, as well as a set of Red Hat keys for verifying packages. To import additional keys for use with **RPM**, see [Section A.3.2.1, “Importing GPG Keys”](#).

A.3.2.1. Importing GPG Keys

To verify Red Hat packages, a Red Hat **GPG** key needs to be installed. A set of basic keys is installed by default. To view a list of installed keys, execute the following command at a shell prompt:

```
~]$ rpm -qa gpg-pubkey*
```

To display details about a specific key, use **rpm -qi** followed by the output from the previous command. For example:

```
~]$ rpm -qi gpg-pubkey-fd431d51-4ae0493b
```

Use the **rpmkeys** command with the **--import** option to install a new key for use with **RPM**. The default location for storing **RPM** GPG keys is the **/etc/pki/rpm-gpg/** directory. To import new keys, use a command like the following as the **root** user:

```
~]# rpmkeys --import /etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
```

See the [Product Signing \(GPG\) Keys](#) article on the Red Hat Customer Portal for additional information about Red Hat package-signing practices.

A.4. Practical and Common Examples of RPM Usage

RPM is a useful tool for both managing your system and diagnosing and fixing problems. See the following examples for an overview of some of the most-used options.

- ▶ To verify your entire system and see what files are missing, issue the following command as **root**:

```
rpm -Va
```

If some files are missing or appear corrupted, consider reinstalling relevant packages.

- ▶ To determine which package owns a file, enter:

```
rpm -qf file
```

- ▶ To verify the package that owns a particular file, enter as **root**:

```
rpm -Vf file
```

- ▶ To locate documentation files that are a part of a package to which a file belongs, enter:

```
rpm -qdf file
```

- ▶ To find information about a (non-installed) package file, use the following command:

```
rpm -qip package.rpm
```

- ▶ To list files contained in a package, use:

```
rpm -qlp package.rpm
```

See the **rpm(8)** manual page for more options.

A.5. Additional Resources

RPM is a complex utility with many options and methods for querying, installing, upgrading, and removing packages. See the following resources to learn more about **RPM**.

Installed Documentation

- ▶ **rpm --help** — This command displays a quick reference of **RPM** parameters.
- ▶ **rpm(8)** — The **RPM** manual page offers an overview of all available **RPM** parameters.

Online Documentation

- ▶ [Red Hat Enterprise Linux 7 Security Guide](#) — The *Security Guide* for Red Hat Enterprise Linux 7 documents how to keep your system up-to-date using the **Yum** package manager and how to verify and install downloaded packages.
- ▶ The **RPM** website — <http://www.rpm.org/>
- ▶ The **RPM** mailing list — <http://lists.rpm.org/mailman/listinfo/rpm-list>

See Also

- ▶ [Chapter 5, Yum](#) describes how to use the **Yum** package manager to search, install, update, and uninstall packages on the command line.

The X Window System

While the heart of Red Hat Enterprise Linux is the kernel, for many users, the face of the operating system is the graphical environment provided by the *X Window System*, also called *X*.

Other windowing environments have existed in the UNIX world, including some that predate the release of the X Window System in June 1984. Nonetheless, *X* has been the default graphical environment for most UNIX-like operating systems, including Red Hat Enterprise Linux, for many years.

The graphical environment for Red Hat Enterprise Linux is supplied by the *X.Org Foundation*, an open source organization created to manage development and strategy for the X Window System and related technologies. *X.Org* is a large-scale, rapid-developing project with hundreds of developers around the world. It features a wide degree of support for a variety of hardware devices and architectures, and runs on myriad operating systems and platforms.

The X Window System uses a client-server architecture. Its main purpose is to provide network transparent window system, which runs on a wide range of computing and graphics machines. The *X server* (the **Xorg** binary) listens for connections from *X client* applications via a network or local loopback interface. The server communicates with the hardware, such as the video card, monitor, keyboard, and mouse. *X client* applications exist in the user space, creating a *graphical user interface (GUI)* for the user and passing user requests to the *X server*.

B.1. The X Server

Red Hat Enterprise Linux 7 uses X server version, which includes several video drivers, EXA, and platform support enhancements over the previous release, among others. In addition, this release includes several automatic configuration features for the X server, as well as the generic input driver, **evdev**, that supports all input devices that the kernel knows about, including most mice and keyboards.

X11R7.1 was the first release to take specific advantage of making the X Window System modular. This release split X into logically distinct modules, which make it easier for open source developers to contribute code to the system.

In the current release, all libraries, headers, and binaries live under the **/usr/** directory. The **/etc/X11/** directory contains configuration files for X client and server applications. This includes configuration files for the X server itself, the X display managers, and many other base components.

The configuration file for the newer Fontconfig-based font architecture is still **/etc/fonts/fonts.conf**. For more information on configuring and adding fonts, see [Section B.4, “Fonts”](#).

Because the X server performs advanced tasks on a wide array of hardware, it requires detailed information about the hardware it works on. The X server is able to automatically detect most of the hardware that it runs on and configure itself accordingly. Alternatively, hardware can be manually specified in configuration files.

The Red Hat Enterprise Linux system installer, Anaconda, installs and configures X automatically, unless the X packages are not selected for installation. If there are any changes to the monitor, video card or other devices managed by the X server, most of the time, X detects and reconfigures these changes automatically. In rare cases, X must be reconfigured manually.

B.2. Desktop Environments and Window Managers

Once an X server is running, X client applications can connect to it and create a GUI for the user. A range of GUIs are available with Red Hat Enterprise Linux, from a standalone window manager, such as *metacity*, to the highly developed and interactive desktop environments *GNOME* and *KDE*, that most Red Hat Enterprise Linux users are familiar with.

To create the latter, more comprehensive GUI, two main classes of X client application must connect to the X server: a *window manager* and a *desktop environment*.

B.2.1. Desktop Environments

A desktop environment integrates various X clients to create a common graphical user environment and a development platform.

Desktop environments have advanced features allowing X clients and other running processes to communicate with one another, while also allowing all applications written to work in that environment to perform advanced tasks, such as drag-and-drop operations.

Red Hat Enterprise Linux provides two desktop environments:

- ▶ *GNOME* — The default desktop environment for Red Hat Enterprise Linux based on the GTK+ 3 graphical toolkit.
- ▶ *KDE* — An alternative desktop environment based on the Qt 4 graphical toolkit.

Both GNOME and KDE have advanced-productivity applications, such as word processors, spreadsheets, and Web browsers; both also provide tools to customize the look and feel of the GUI. Additionally, if both the GTK+ 3 and the Qt libraries are present, KDE applications can run in GNOME and vice versa.

B.2.2. Window Managers

Window managers are X client programs which are either part of a desktop environment or, in some cases, stand-alone. Their primary purpose is to control the way graphical windows are positioned, resized, or moved. Window managers also control title bars, window focus behavior, and user-specified key and mouse button bindings.

The Red Hat Enterprise Linux repositories provide several different window managers.

gnome-shell

GNOME Shell is the the default window manager for GNOME. It provides the user experience of GNOME and is automatically pulled in as a dependency when the GNOME desktop is installed.

metacity

The *Metacity* window manager is a simple and efficient window manager that can be used when a full desktop environment is not needed.

kwin

The *KWin* window manager is the default window manager for KDE. It is an efficient window manager which supports custom themes. This window manager is automatically pulled in as a dependency when the KDE desktop is installed.

mwm

The *Motif Window Manager* (**mwm**) is a basic, stand-alone window manager. Since it is designed to be stand-alone, it should not be used in conjunction with GNOME or KDE. To run this window manager, you need to install the *openmotif* package.

B.3. X Server Configuration Files

The X server is a single binary executable `/usr/bin/Xorg`; a symbolic link `X` pointing to this file is also provided. Associated configuration files are stored in the `/etc/X11/` and `/usr/share/X11/` directories.

The X Window System supports two different configuration schemes. Configuration files in the `xorg.conf.d` directory contain preconfigured settings from vendors and from distribution, and these files should not be edited by hand. Configuration in the `xorg.conf` file, on the other hand, is done completely by hand but is not necessary in most scenarios.



When do you need the `xorg.conf` file?

All necessary parameters for a display and peripherals are auto-detected and configured during installation. The configuration file for the X server, `/etc/X11/xorg.conf`, that was necessary in previous releases, is not supplied with the current release of the X Window System. It can still be useful to create the file manually to configure new hardware, to set up an environment with multiple video cards, or for debugging purposes.

The `/usr/lib/xorg/modules/` (or `/usr/lib64/xorg/modules/`) directory contains X server modules that can be loaded dynamically at runtime. By default, only some modules in `/usr/lib/xorg/modules/` are automatically loaded by the X server.

When Red Hat Enterprise Linux 7 is installed, the configuration files for X are created using information gathered about the system hardware during the installation process by the HAL (Hardware Abstraction Layer) configuration back end. Whenever the X server is started, it asks HAL for the list of input devices and adds each of them with their respective driver. Whenever a new input device is plugged in, or an existing input device is removed, HAL notifies the X server about the change. Because of this notification system, devices using the `mouse`, `kbd`, or `vmmouse` driver configured in the `xorg.conf` file are, by default, ignored by the X server. See [Section B.3.3.3, “The `ServerFlags` section”](#) for further details. Additional configuration is provided in the `/etc/X11/xorg.conf.d/` directory and it can override or augment any configuration that has been obtained through HAL.

B.3.1. The Structure of the Configuration

The format of the X configuration files is comprised of many different sections which address specific aspects of the system hardware. Each section begins with a **Section "section-name"** line, where "section-name" is the title for the section, and ends with an **EndSection** line. Each section contains lines that include option names and one or more option values. Some of these are sometimes enclosed in double quotes ("").

Some options within the `/etc/X11/xorg.conf` file accept a Boolean switch which turns the feature on or off. The acceptable values are:

- ▶ **1, on, true, or yes** — Turns the option on.
- ▶ **0, off, false, or no** — Turns the option off.

The following shows a typical configuration file for the keyboard. Lines beginning with a hash sign (#) are not read by the X server and are used for human-readable comments.

```
# This file is autogenerated by system-setup-keyboard. Any
# modifications will be lost.

Section "InputClass"
```

```

Identifier "system-setup-keyboard"
MatchIsKeyboard "on"
Option "XkbModel" "pc105"
Option "XkbLayout" "cz,us"
# Option "XkbVariant" "(null)"
Option "XkbOptions" "terminate:ctrl_alt_bksp,grp:shifts_toggle,grp_led:scroll"
EndSection

```

B.3.2. The `xorg.conf.d` Directory

The X server supports two configuration directories. The `/usr/share/X11/xorg.conf.d/` provides separate configuration files from vendors or third-party packages; changes to files in this directory may be overwritten by settings specified in the `/etc/X11/xorg.conf` file. The `/etc/X11/xorg.conf.d/` directory stores user-specific configuration.

Files with the suffix `.conf` in configuration directories are parsed by the X server upon startup and are treated like part of the traditional `xorg.conf` configuration file. These files may contain one or more sections; for a description of the options in a section and the general layout of the configuration file, see [Section B.3.3, “The `xorg.conf` File”](#) or to the `xorg.conf(5)` man page. The X server essentially treats the collection of configuration files as one big file with entries from `xorg.conf` at the end. Users are encouraged to put custom configuration into `/etc/xorg.conf` and leave the directory for configuration snippets provided by the distribution.

B.3.3. The `xorg.conf` File

In previous releases of the X Window System, `/etc/X11/xorg.conf` file was used to store initial setup for X. When a change occurred with the monitor, video card or other device managed by the X server, the file needed to be edited manually. In Red Hat Enterprise Linux, there is rarely a need to manually create and edit the `/etc/X11/xorg.conf` file. Nevertheless, it is still useful to understand various sections and optional parameters available, especially when troubleshooting or setting up unusual hardware configuration.

In the following, some important sections are described in the order in which they appear in a typical `/etc/X11/xorg.conf` file. More detailed information about the X server configuration file can be found in the `xorg.conf(5)` man page. This section is mostly intended for advanced users as most configuration options described below are not needed in typical configuration scenarios.

B.3.3.1. The `InputClass` section

InputClass is a new type of configuration section that does not apply to a single device but rather to a class of devices, including hot-plugged devices. An **InputClass** section's scope is limited by the matches specified; in order to apply to an input device, all matches must apply to the device as seen in the example below:

```

Section "InputClass"
    Identifier      "touchpad catchall"
    MatchIsTouchpad "on"
    Driver         "synaptics"
EndSection

```

If this snippet is present in an `xorg.conf` file or an `xorg.conf.d` directory, any touchpad present in the system is assigned the `synaptics` driver.



Alphanumeric sorting in `xorg.conf.d`

Note that due to alphanumeric sorting of configuration files in the `xorg.conf.d` directory, the **Driver** setting in the example above overwrites previously set driver options. The more generic the class, the earlier it should be listed.

The match options specify which devices a section may apply to. To match a device, all match options must correspond. The following options are commonly used in the **InputClass** section:

- ▶ **MatchIsPointer**, **MatchIsKeyboard**, **MatchIsTouchpad**, **MatchIsTouchscreen**, **MatchIsJoystick** — Boolean options to specify a type of a device.
- ▶ **MatchProduct "product_name"** — this option matches if the *product_name* substring occurs in the product name of the device.
- ▶ **MatchVendor "vendor_name"** — this option matches if the *vendor_name* substring occurs in the vendor name of the device.
- ▶ **MatchDevicePath "/path/to/device"** — this option matches any device if its device path corresponds to the patterns given in the "/path/to/device" template, for example `/dev/input/event*`. See the **fnmatch(3)** man page for further details.
- ▶ **MatchTag "tag_pattern"** — this option matches if at least one tag assigned by the HAL configuration back end matches the *tag_pattern* pattern.

A configuration file may have multiple **InputClass** sections. These sections are optional and are used to configure a class of input devices as they are automatically added. An input device can match more than one **InputClass** section. When arranging these sections, it is recommended to put generic matches above specific ones because each input class can override settings from a previous one if an overlap occurs.

B.3.3.2. The **InputDevice** section

Each **InputDevice** section configures one input device for the X server. Previously, systems typically had at least one **InputDevice** section for the keyboard, and most mouse settings were automatically detected.

With Red Hat Enterprise Linux 7, no **InputDevice** configuration is needed for most setups, and the `xorg-x11-drv-*` input driver packages provide the automatic configuration through HAL. The default driver for both keyboards and mice is **evdev**.

The following example shows a typical **InputDevice** section for a keyboard:

```
Section "InputDevice"
    Identifier "Keyboard0"
    Driver "kbd"
    Option "XkbModel" "pc105"
    Option "XkbLayout" "us"
EndSection
```

The following entries are commonly used in the **InputDevice** section:

- ▶ **Identifier** — Specifies a unique name for this **InputDevice** section. This is a required entry.

▶ **Driver** — Specifies the name of the device driver X must load for the device. If the **AutoAddDevices** option is enabled (which is the default setting), any input device section with **Driver "mouse"** or **Driver "kbd"** will be ignored. This is necessary due to conflicts between the legacy mouse and keyboard drivers and the new **evdev** generic driver. Instead, the server will use the information from the back end for any input devices. Any custom input device configuration in the **xorg.conf** should be moved to the back end. In most cases, the back end will be HAL and the configuration location will be the **/etc/X11/xorg.conf.d** directory.

▶ **Option** — Specifies necessary options pertaining to the device.

A mouse may also be specified to override any auto-detected values for the device. The following options are typically included when adding a mouse in the **xorg.conf** file:

- **Protocol** — Specifies the protocol used by the mouse, such as **IMPS/2**.
- **Device** — Specifies the location of the physical device.
- **Emulate3Buttons** — Specifies whether to allow a two-button mouse to act like a three-button mouse when both mouse buttons are pressed simultaneously.

Consult the **xorg.conf(5)** man page for a complete list of valid options for this section.

B.3.3.3. The ServerFlags section

The optional **ServerFlags** section contains miscellaneous global X server settings. Any settings in this section may be overridden by options placed in the **ServerLayout** section (refer to [Section B.3.3.4, “The ServerLayout Section”](#) for details).

Each entry within the **ServerFlags** section occupies a single line and begins with the term **Option** followed by an option enclosed in double quotation marks ("").

The following is a sample **ServerFlags** section:

```
Section "ServerFlags"
    Option "DontZap" "true"
EndSection
```

The following lists some of the most useful options:

▶ **"DontZap" "boolean"** — When the value of <boolean> is set to **true**, this setting prevents the use of the **Ctrl+Alt+Backspace** key combination to immediately terminate the X server.

X keyboard extension

Even if this option is enabled, the key combination still must be configured in the X Keyboard Extension (XKB) map before it can be used. One way how to add the key combination to the map is to run the following command:

```
setxkbmap -option "terminate:ctrl_alt_bksp"
```

▶ **"DontZoom" "boolean"** — When the value of <boolean> is set to **true**, this setting prevents cycling through configured video resolutions using the **Ctrl+Alt+Keypad-Plus** and **Ctrl+Alt+Keypad-Minus** key combinations.

- ▶ **"AutoAddDevices" "boolean"** — When the value of <boolean> is set to **false**, the server will not hot plug input devices and instead rely solely on devices configured in the **xorg.conf** file. See [Section B.3.3.2, “The InputDevice section”](#) for more information concerning input devices. This option is enabled by default and HAL (hardware abstraction layer) is used as a back end for device discovery.

B.3.3.4. The ServerLayout Section

The **ServerLayout** section binds together the input and output devices controlled by the X server. At a minimum, this section must specify one input device and one output device. By default, a monitor (output device) and a keyboard (input device) are specified.

The following example shows a typical **ServerLayout** section:

```
Section "ServerLayout"
    Identifier "Default Layout"
    Screen 0 "Screen0" 0 0
    InputDevice "Mouse0" "CorePointer"
    InputDevice "Keyboard0" "CoreKeyboard"
EndSection
```

The following entries are commonly used in the **ServerLayout** section:

- ▶ **Identifier** — Specifies a unique name for this **ServerLayout** section.
- ▶ **Screen** — Specifies the name of a **Screen** section to be used with the X server. More than one **Screen** option may be present.

The following is an example of a typical **Screen** entry:

```
Screen 0 "Screen0" 0 0
```

The first number in this example **Screen** entry (**0**) indicates that the first monitor connector, or *head* on the video card, uses the configuration specified in the **Screen** section with the identifier **"Screen0"**.

An example of a **Screen** section with the identifier **"Screen0"** can be found in [Section B.3.3.8, “The Screen section”](#).

If the video card has more than one head, another **Screen** entry with a different number and a different **Screen** section identifier is necessary.

The numbers to the right of **"Screen0"** give the absolute X and Y coordinates for the upper left corner of the screen (**0 0** by default).

- ▶ **InputDevice** — Specifies the name of an **InputDevice** section to be used with the X server.

It is advisable that there be at least two **InputDevice** entries: one for the default mouse and one for the default keyboard. The options **CorePointer** and **CoreKeyboard** indicate that these are the primary mouse and keyboard. If the **AutoAddDevices** option is enabled, this entry needs not to be specified in the **ServerLayout** section. If the **AutoAddDevices** option is disabled, both mouse and keyboard are auto-detected with the default values.

- ▶ **Option "option-name"** — An optional entry which specifies extra parameters for the section. Any options listed here override those listed in the **ServerFlags** section.

Replace *<option-name>* with a valid option listed for this section in the **xorg.conf(5)** man page.

It is possible to put more than one **ServerLayout** section in the **/etc/X11/xorg.conf** file. By default, the server only reads the first one it encounters, however. If there is an alternative **ServerLayout** section, it can be specified as a command line argument when starting an X session; as in the **Xorg - layout <layoutname>** command.

B.3.3.5. The Files section

The **Files** section sets paths for services vital to the X server, such as the font path. This is an optional section, as these paths are normally detected automatically. This section can be used to override automatically detected values.

The following example shows a typical **Files** section:

```
Section "Files"
    RgbPath "/usr/share/X11/rgb.txt"
    FontPath "unix/:7100"
EndSection
```

The following entries are commonly used in the **Files** section:

- ▶ **ModulePath** — An optional parameter which specifies alternate directories which store X server modules.

B.3.3.6. The Monitor section

Each **Monitor** section configures one type of monitor used by the system. This is an optional entry as most monitors are now detected automatically.

This example shows a typical **Monitor** section for a monitor:

```
Section "Monitor"
    Identifier "Monitor0"
    VendorName "Monitor Vendor"
    ModelName "DDC Probed Monitor - ViewSonic G773-2"
    DisplaySize 320 240
    HorizSync 30.0 - 70.0
    VertRefresh 50.0 - 180.0
EndSection
```

The following entries are commonly used in the **Monitor** section:

- ▶ **Identifier** — Specifies a unique name for this **Monitor** section. This is a required entry.
- ▶ **VendorName** — An optional parameter which specifies the vendor of the monitor.
- ▶ **ModelName** — An optional parameter which specifies the monitor's model name.
- ▶ **DisplaySize** — An optional parameter which specifies, in millimeters, the physical size of the monitor's picture area.
- ▶ **HorizSync** — Specifies the range of horizontal sync frequencies compatible with the monitor, in kHz. These values help the X server determine the validity of built-in or specified **Modeline** entries for the monitor.
- ▶ **VertRefresh** — Specifies the range of vertical refresh frequencies supported by the monitor, in kHz.

These values help the X server determine the validity of built-in or specified **Modeline** entries for the monitor.

- ▶ **Modeline** — An optional parameter which specifies additional video modes for the monitor at particular resolutions, with certain horizontal sync and vertical refresh resolutions. See the **xorg.conf(5)** man page for a more detailed explanation of **Modeline** entries.
- ▶ **Option "option-name"** — An optional entry which specifies extra parameters for the section. Replace *<option-name>* with a valid option listed for this section in the **xorg.conf(5)** man page.

B.3.3.7. The Device section

Each **Device** section configures one video card on the system. While one **Device** section is the minimum, additional instances may occur for each video card installed on the machine.

The following example shows a typical **Device** section for a video card:

```
Section "Device"
    Identifier "Videocard0"
    Driver "mga"
    VendorName "Videocard vendor"
    BoardName "Matrox Millennium G200"
    VideoRam 8192
    Option "dpms"
EndSection
```

The following entries are commonly used in the **Device** section:

- ▶ **Identifier** — Specifies a unique name for this **Device** section. This is a required entry.
- ▶ **Driver** — Specifies which driver the X server must load to utilize the video card. A list of drivers can be found in **/usr/share/hwdata/videodrivers**, which is installed with the **hwdata** package.
- ▶ **VendorName** — An optional parameter which specifies the vendor of the video card.
- ▶ **BoardName** — An optional parameter which specifies the name of the video card.
- ▶ **VideoRam** — An optional parameter which specifies the amount of RAM available on the video card, in kilobytes. This setting is only necessary for video cards the X server cannot probe to detect the amount of video RAM.
- ▶ **BusID** — An entry which specifies the bus location of the video card. On systems with only one video card a **BusID** entry is optional and may not even be present in the default **/etc/X11/xorg.conf** file. On systems with more than one video card, however, a **BusID** entry is required.
- ▶ **Screen** — An optional entry which specifies which monitor connector or head on the video card the **Device** section configures. This option is only useful for video cards with multiple heads.

If multiple monitors are connected to different heads on the same video card, separate **Device** sections must exist and each of these sections must have a different **Screen** value.

Values for the **Screen** entry must be an integer. The first head on the video card has a value of **0**. The value for each additional head increments this value by one.

- ▶ **Option "option-name"** — An optional entry which specifies extra parameters for the section. Replace *<option-name>* with a valid option listed for this section in the **xorg.conf(5)** man page.

One of the more common options is "**dpm s**" (for Display Power Management Signaling, a VESA standard), which activates the Energy Star energy compliance setting for the monitor.

B.3.3.8. The Screen section

Each **Screen** section binds one video card (or video card head) to one monitor by referencing the **Device** section and the **Monitor** section for each. While one **Screen** section is the minimum, additional instances may occur for each video card and monitor combination present on the machine.

The following example shows a typical **Screen** section:

```
Section "Screen"
    Identifier "Screen0"
    Device "Videocard0"
    Monitor "Monitor0"
    DefaultDepth 16

    SubSection "Display"
        Depth 24
        Modes "1280x1024" "1280x960" "1152x864" "1024x768" "800x600" "640x480"
    EndSubSection

    SubSection "Display"
        Depth 16
        Modes "1152x864" "1024x768" "800x600" "640x480"
    EndSubSection
EndSection
```

The following entries are commonly used in the **Screen** section:

- ▶ **Identifier** — Specifies a unique name for this **Screen** section. This is a required entry.
- ▶ **Device** — Specifies the unique name of a **Device** section. This is a required entry.
- ▶ **Monitor** — Specifies the unique name of a **Monitor** section. This is only required if a specific **Monitor** section is defined in the **xorg.conf** file. Normally, monitors are detected automatically.
- ▶ **DefaultDepth** — Specifies the default color depth in bits. In the previous example, **16** (which provides thousands of colors) is the default. Only one **DefaultDepth** entry is permitted, although this can be overridden with the Xorg command line option **-depth <n>**, where **<n>** is any additional depth specified.
- ▶ **SubSection "Display"** — Specifies the screen modes available at a particular color depth. The **Screen** section can have multiple **Display** subsections, which are entirely optional since screen modes are detected automatically.

This subsection is normally used to override auto-detected modes.

- ▶ **Option "<option-name>"** — An optional entry which specifies extra parameters for the section. Replace **<option-name>** with a valid option listed for this section in the **xorg.conf(5)** man page.

B.3.3.9. The DRI section

The optional **DRI** section specifies parameters for the *Direct Rendering Infrastructure (DRI)*. DRI is an interface which allows 3D software applications to take advantage of 3D hardware acceleration capabilities built into most modern video hardware. In addition, DRI can improve 2D performance via hardware acceleration, if supported by the video card driver.

This section is rarely used, as the DRI Group and Mode are automatically initialized to default values. If a different Group or Mode is needed, then adding this section to the **xorg.conf** file will override the default values.

The following example shows a typical **DRI** section:

```
Section "DRI"
    Group 0
    Mode 0666
EndSection
```

Since different video cards use DRI in different ways, do not add to this section without first referring to <http://dri.freedesktop.org/wiki/>.

B.4. Fonts

Red Hat Enterprise Linux uses *Fontconfig* subsystem to manage and display fonts under the X Window System. It simplifies font management and provides advanced display features, such as anti-aliasing. This system is used automatically for applications programmed using the **Qt 3** or **GTK+ 2** graphical toolkits, or their newer versions.

The Fontconfig font subsystem allows applications to directly access fonts on the system and use the *XFreeType interface library (Xft)* or other rendering mechanisms to render Fontconfig fonts with advanced features such as anti-aliasing. Graphical applications can use the Xft library with Fontconfig to draw text to the screen.

Font configuration

Fontconfig uses the **/etc/fonts/fonts.conf** configuration file, which should not be edited by hand.

FONTs group

Any system where the user expects to run remote X applications needs to have the **fonts** group installed. This can be done by selecting the group in the installer, and also by running the **yum groupinstall fonts** command after installation.

B.4.1. Adding Fonts to Fontconfig

Adding new fonts to the Fontconfig subsystem is a straightforward process:

1. To add fonts for an individual user, copy the new fonts into the **.fonts/** directory in the user's home directory.

To add fonts system-wide, copy the new fonts into the **/usr/share/fonts/** directory. It is a good idea to create a new subdirectory, such as **local/** or similar, to help distinguish between user-installed and default fonts.

2. Run the **fc-cache** command as root to update the font information cache:

```
fc-cache <path-to-font-directory>
```

In this command, replace `<path-to-font-directory>` with the directory containing the new fonts (either `/usr/share/fonts/local/` or `/home/<user>/ .fonts/`).



Interactive font installation

Individual users may also install fonts interactively, by typing `fonts:///` into the **Nautilus** address bar, and dragging the new font files there.

B.5. Runlevels, targets, and X

In Red Hat Enterprise Linux 7, the concept of runlevels has been replaced with *systemd targets*. For more information about targets, see [Chapter 6, Managing Services with systemd](#).

B.6. Additional Resources

There is a large amount of detailed information available about the X server, the clients that connect to it, and the assorted desktop environments and window managers.

B.6.1. Installed Documentation

- ▶ `/usr/share/X11/doc/` — contains detailed documentation on the X Window System architecture, as well as how to get additional information about the Xorg project as a new user.
- ▶ `/usr/share/doc/gdm-<version-number>/README` — contains information on how display managers control user authentication.
- ▶ `man xorg.conf` — Contains information about the `xorg.conf` configuration files, including the meaning and syntax for the different sections within the files.
- ▶ `man Xorg` — Describes the `Xorg` display server.

B.6.2. Useful Websites

- ▶ <http://www.X.org/> — Home page of the X.Org Foundation, which produces major releases of the X Window System bundled with Red Hat Enterprise Linux to control the necessary hardware and provide a GUI environment.
- ▶ <http://dri.sourceforge.net/> — Home page of the DRI (Direct Rendering Infrastructure) project. The DRI is the core hardware 3D acceleration component of X.
- ▶ <http://www.gnome.org/> — Home of the GNOME project.
- ▶ <http://www.kde.org/> — Home of the KDE desktop environment.

Revision History

Revision 0.11-2

Fri 25 Jun 2014

Stephen Wadeley

Red Hat Enterprise Linux 7.0 GA release of the System Administrator's Guide.

C.1. Acknowledgments

Certain portions of this text first appeared in the >*Red Hat Enterprise Linux 6 Deployment Guide*, copyright © 2014 Red Hat, Inc., available at https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/index.html.

Section 16.6, “Monitoring Performance with Net-SNMP” is based on an article written by Michael Solberg.

Index

Symbols

.fetchmailrc, Fetchmail Configuration Options

- server options, [Server Options](#)
- user options, [User Options](#)

.procmailrc, Procmail Configuration

/dev/oprofile/, Understanding /dev/oprofile/

/var/spool/anacron , Configuring Anacron Jobs

/var/spool/cron , Configuring Cron Jobs

(see OProfile)

A

ABRT, Introduction to ABRT

- (see also abrtd)
- (see also Bugzilla)
- (see also Red Hat Technical Support)
- additional resources, [Additional Resources](#)
- autoreporting, [Setting Up Automatic Reporting](#)
- CLI, [Using the Command Line Tool](#)
- configuring, [Configuring ABRT](#)
- configuring events, [Configuring Events](#)
- crash detection, [Introduction to ABRT](#)
- creating events, [Creating Custom Events](#)
- GUI, [Using the GUI](#)
- installing, [Installing ABRT and Starting its Services](#)
- introducing, [Introduction to ABRT](#)
- problems
 - detecting, [Detecting Software Problems](#)
 - handling of, [Handling Detected Problems](#)
 - supported, [Detecting Software Problems](#)
- standard events, [Configuring Events](#)
- starting, [Installing ABRT and Starting its Services](#), [Starting the ABRT Services](#)
- testing, [Testing ABRT Crash Detection](#)

ABRT CLI

- installing, [Installing ABRT for the Command Line](#)

ABRT GUI

- installing, [Installing the ABRT GUI](#)

ABRT Tools

- installing, [Installing Supplementary ABRT Tools](#)

abrtd

- additional resources, [Additional Resources](#)
- restarting, [Starting the ABRT Services](#)
- starting, [Installing ABRT and Starting its Services](#), [Starting the ABRT Services](#)
- status, [Starting the ABRT Services](#)
- testing, [Testing ABRT Crash Detection](#)

adding

- group, [Adding a New Group](#)
- user, [Adding a New User](#)

anacron, Cron and Anacron

- anacron configuration file, [Configuring Anacron Jobs](#)
- user-defined tasks, [Configuring Anacron Jobs](#)

anacrontab , Configuring Anacron Jobs**Apache HTTP Server**

- additional resources
 - installed documentation, [Installed Documentation](#)
 - useful websites, [Useful Websites](#)
- checking configuration, [Editing the Configuration Files](#)
- checking status, [Verifying the Service Status](#)
- directories
 - /etc/httpd/conf.d/ , [Editing the Configuration Files](#)
 - /usr/lib/httpd/modules/ , [Working with Modules](#)
 - /usr/lib64/httpd/modules/ , [Working with Modules](#)
- files
 - /etc/httpd/conf.d/ssl.conf , [Enabling the mod_ssl Module](#)
 - /etc/httpd/conf/httpd.conf , [Editing the Configuration Files](#)
- modules
 - developing, [Writing a Module](#)
 - loading, [Loading a Module](#)
 - mod_ssl , [Setting Up an SSL Server](#)
 - mod_userdir, [Updating the Configuration](#)
- restarting, [Restarting the Service](#)
- SSL server
 - certificate, [An Overview of Certificates and Security](#), [Using an Existing Key and Certificate](#), [Generating a New Key and Certificate](#)
 - certificate authority, [An Overview of Certificates and Security](#)
 - private key, [An Overview of Certificates and Security](#), [Using an Existing Key and Certificate](#), [Generating a New Key and Certificate](#)
 - public key, [An Overview of Certificates and Security](#)

- starting, [Starting the Service](#)
- stopping, [Stopping the Service](#)
- version 2.4
 - changes, [Notable Changes](#)
 - updating from version 2.2, [Updating the Configuration](#)
- virtual host, [Setting Up Virtual Hosts](#)

at , At and Batch

- additional resources, [Additional Resources](#)

Automated Tasks, Automating System Tasks**B****batch , At and Batch**

- additional resources, [Additional Resources](#)

blkid, Using the blkid Command**boot loader**

- GRUB 2 boot loader, [Working with the GRUB 2 Boot Loader](#)
- verifying, [Verifying the Boot Loader](#)

boot media, Preparing to Upgrade**C****ch-email .fetchmailrc**

- global options, [Global Options](#)

Configuration File Changes, Preserving Configuration File Changes**CPU usage, Viewing CPU Usage****createrepo, Creating a Yum Repository****cron, Cron and Anacron**

- additional resources, [Additional Resources](#)
- cron configuration file, [Configuring Cron Jobs](#)
- user-defined tasks, [Configuring Cron Jobs](#)

crontab , Configuring Cron Jobs**CUPS (see Printer Configuration)****D****desktop environments (see X)****df, Using the df Command****directory server (see OpenLDAP)****documentation**

- finding installed, [Practical and Common Examples of RPM Usage](#)

drivers (see kernel module)**du, Using the du Command**

E**ECDSA keys**

- generating, [Generating Key Pairs](#)

email

- additional resources, [Additional Resources](#)
 - installed documentation, [Installed Documentation](#)
 - related books, [Related Books](#)
 - useful websites, [Useful Websites](#)
- Fetchmail, [Fetchmail](#)
- mail server
 - Dovecot, [Dovecot](#)
- Postfix, [Postfix](#)
- Procmail, [Mail Delivery Agents](#)
- program classifications, [Email Program Classifications](#)
- protocols, [Email Protocols](#)
 - IMAP, [IMAP](#)
 - POP, [POP](#)
 - SMTP, [SMTP](#)
- security, [Securing Communication](#)
 - clients, [Secure Email Clients](#)
 - servers, [Securing Email Client Communications](#)
- Sendmail, [Sendmail](#)
- spam
 - filtering out, [Spam Filters](#)
- types
 - Mail Delivery Agent, [Mail Delivery Agent](#)
 - Mail Transport Agent, [Mail Transport Agent](#)
 - Mail User Agent, [Mail User Agent](#)

extra packages for Enterprise Linux (EPEL)

- installable packages, [Finding RPM Packages](#)

F**Fetchmail, [Fetchmail](#)**

- additional resources, [Additional Resources](#)
- command options, [Fetchmail Command Options](#)
 - informational, [Informational or Debugging Options](#)
 - special, [Special Options](#)
- configuration options, [Fetchmail Configuration Options](#)
 - global options, [Global Options](#)
 - server options, [Server Options](#)
 - user options, [User Options](#)

file systems, [Viewing Block Devices and File Systems](#)**findmnt, [Using the findmnt Command](#)****free, [Using the free Command](#)****FTP, [FTP](#)**

- (see also vsftpd)
- active mode, [The File Transfer Protocol](#)
- command port, [The File Transfer Protocol](#)
- data port, [The File Transfer Protocol](#)
- definition of, [FTP](#)
- introducing, [The File Transfer Protocol](#)
- passive mode, [The File Transfer Protocol](#)

G**GNOME, Desktop Environments**

- (see also X)

gnome-system-log (see Log File Viewer)**gnome-system-monitor, Using the System Monitor Tool, Using the System Monitor Tool, Using the System Monitor Tool, Using the System Monitor Tool****GnuPG**

- checking RPM package signatures, [Checking Package Signatures](#)

group configuration

- adding groups, [Adding a New Group](#)
- filtering list of groups, [Viewing Users and Groups](#)
- groupadd, [Adding a New Group](#)
- modify users in groups, [Modifying Group Properties](#)
- modifying group properties, [Modifying Group Properties](#)
- viewing list of groups, [Using the User Manager Tool](#)

groups (see group configuration)

- GID, [Managing Users and Groups](#)
- introducing, [Managing Users and Groups](#)
- shared directories, [Creating Group Directories](#)
- tools for management of
 - groupadd, [User Private Groups, Using Command Line Tools](#)
 - system-config-users, [User Private Groups](#)
 - User Manager, [Using Command Line Tools](#)
- user private, [User Private Groups](#)

GRUB 2

- configuring GRUB 2, [Working with the GRUB 2 Boot Loader](#)
- customizing GRUB 2, [Working with the GRUB 2 Boot Loader](#)
- reinstalling GRUB 2, [Working with the GRUB 2 Boot Loader](#)

H**hardware**

- viewing, [Viewing Hardware Information](#)

HTTP server (see Apache HTTP Server)**httpd (see Apache HTTP Server)****I****information**

- about your system, [System Monitoring Tools](#)

initial RAM disk image

- verifying, [Verifying the Initial RAM Disk Image](#)
- IBM eServer System i, [Verifying the Initial RAM Disk Image](#)

initial RPM repositories

- installable packages, [Finding RPM Packages](#)

insmod, [Loading a Module](#)

- (see also kernel module)

installing the kernel, [Manually Upgrading the Kernel](#)**K****KDE, [Desktop Environments](#)**

- (see also X)

kernel

- downloading, [Downloading the Upgraded Kernel](#)
- installing kernel packages, [Manually Upgrading the Kernel](#)
- kernel packages, [Overview of Kernel Packages](#)
- package, [Manually Upgrading the Kernel](#)
- performing kernel upgrade, [Performing the Upgrade](#)
- RPM package, [Manually Upgrading the Kernel](#)
- upgrade kernel available, [Downloading the Upgraded Kernel](#)
 - Security Errata, [Downloading the Upgraded Kernel](#)
 - via Red Hat network, [Downloading the Upgraded Kernel](#)
- upgrading
 - preparing, [Preparing to Upgrade](#)
 - working boot media, [Preparing to Upgrade](#)
- upgrading the kernel, [Manually Upgrading the Kernel](#)

kernel module

- definition, [Working with Kernel Modules](#)
- directories
 - /etc/sysconfig/modules/, [Persistent Module Loading](#)
 - /lib/modules/<kernel_version>/kernel/drivers/, [Loading a Module](#)
- files
 - /proc/modules, [Listing Currently-Loaded Modules](#)
- listing
 - currently loaded modules, [Listing Currently-Loaded Modules](#)
 - module information, [Displaying Information About a Module](#)
- loading
 - at the boot time, [Persistent Module Loading](#)
 - for the current session, [Loading a Module](#)
- module parameters
 - supplying, [Setting Module Parameters](#)
- unloading, [Unloading a Module](#)
- utilities
 - insmod, [Loading a Module](#)

- lsmod, [Listing Currently-Loaded Modules](#)
- modinfo, [Displaying Information About a Module](#)
- modprobe, [Loading a Module, Unloading a Module](#)
- rmmod, [Unloading a Module](#)

kernel package

- kernel
 - for single,multicore and multiprocessor systems, [Overview of Kernel Packages](#)
- kernel-devel
 - kernel headers and makefiles, [Overview of Kernel Packages](#)
- kernel-headers
 - C header files files, [Overview of Kernel Packages](#)
- linux-firmware
 - firmware files, [Overview of Kernel Packages](#)
- perf
 - firmware files, [Overview of Kernel Packages](#)

kernel upgrading

- preparing, [Preparing to Upgrade](#)

keyboard configuration, [System Locale and Keyboard Configuration](#)

- layout, [Changing the Keyboard Layout](#)

kwin, [Window Managers](#)

- (see also X)

L**LDAP (see OpenLDAP)****localectl (see keyboard configuration)****Log File Viewer**

- filtering, [Viewing Log Files](#)
- monitoring, [Monitoring Log Files](#)
- refresh rate, [Viewing Log Files](#)
- searching, [Viewing Log Files](#)

log files, [Viewing and Managing Log Files](#)

- (see also Log File Viewer)
- description, [Viewing and Managing Log Files](#)
- locating, [Locating Log Files](#)
- monitoring, [Monitoring Log Files](#)
- rotating, [Locating Log Files](#)
- rsyslogd daemon, [Viewing and Managing Log Files](#)
- viewing, [Viewing Log Files](#)

logrotate, [Locating Log Files](#)**lsblk, [Using the lsblk Command](#)****lscpu, [Using the lscpu Command](#)****lsmod, [Listing Currently-Loaded Modules](#)**

- (see also kernel module)

Ispci, [Using the Ispci Command](#)

Ispcmcia, [Using the Ispcmcia Command](#)

Isusb, [Using the Isusb Command](#)

M

Mail Delivery Agent (see [email](#))

Mail Transport Agent (see [email](#)) (see [MTA](#))

Mail Transport Agent Switcher, [Mail Transport Agent \(MTA\) Configuration](#)

Mail User Agent, [Mail Transport Agent \(MTA\) Configuration](#) (see [email](#))

MDA (see [Mail Delivery Agent](#))

memory usage, [Viewing Memory Usage](#)

metacity, [Window Managers](#)

- (see also [X](#))

modinfo, [Displaying Information About a Module](#)

- (see also [kernel module](#))

modprobe, [Loading a Module, Unloading a Module](#)

- (see also [kernel module](#))

module (see [kernel module](#))

module parameters (see [kernel module](#))

MTA (see [Mail Transport Agent](#))

- setting default, [Mail Transport Agent \(MTA\) Configuration](#)

- switching with Mail Transport Agent Switcher, [Mail Transport Agent \(MTA\) Configuration](#)

MUA, [Mail Transport Agent \(MTA\) Configuration](#) (see [Mail User Agent](#))

mwm, [Window Managers](#)

- (see also [X](#))

N

net program, [Samba Distribution Programs](#)

nmblookup program, [Samba Distribution Programs](#)

O

opannotate (see [OProfile](#))

opcontrol (see [OProfile](#))

OpenLDAP

- checking status, [Verifying the Service Status](#)

- client applications, [Overview of Common LDAP Client Applications](#)

- configuration

- database, [Changing the Database-Specific Configuration](#)

- global, [Changing the Global Configuration](#)

- overview, [OpenLDAP Server Setup](#)

- directives

- olcAllows, [Changing the Global Configuration](#)

- olcConnMaxPending, [Changing the Global Configuration](#)

- olcConnMaxPendingAuth, [Changing the Global Configuration](#)
- olcDisallow, [Changing the Global Configuration](#)
- olcIdleTimeout, [Changing the Global Configuration](#)
- olcLogFile, [Changing the Global Configuration](#)
- olcReadOnly, [Changing the Database-Specific Configuration](#)
- olcReferral, [Changing the Global Configuration](#)
- olcRootDN, [Changing the Database-Specific Configuration](#)
- olcRootPW, [Changing the Database-Specific Configuration](#)
- olcSuffix, [Changing the Database-Specific Configuration](#)
- olcWriteTimeout, [Changing the Global Configuration](#)

- directories
 - /etc/openldap/slapd.d/, [Configuring an OpenLDAP Server](#)
 - /etc/openldap/slapd.d/cn=config/cn=schema/, [Extending Schema](#)

- features, [OpenLDAP Features](#)
- files
 - /etc/openldap/ldap.conf, [Configuring an OpenLDAP Server](#)
 - /etc/openldap/slapd.d/cn=config.ldif, [Changing the Global Configuration](#)
 - /etc/openldap/slapd.d/cn=config/olcDatabase={1}bdb.ldif, [Changing the Database-Specific Configuration](#)

- installation, [Installing the OpenLDAP Suite](#)
- migrating authentication information, [Migrating Old Authentication Information to LDAP Format](#)
- packages, [Installing the OpenLDAP Suite](#)
- restarting, [Restarting the Service](#)
- running, [Starting the Service](#)
- schema, [Extending Schema](#)
- stopping, [Stopping the Service](#)
- terminology
 - attribute, [LDAP Terminology](#)
 - entry, [LDAP Terminology](#)
 - LDIF, [LDAP Terminology](#)

- utilities, [Overview of OpenLDAP Server Utilities](#), [Overview of OpenLDAP Client Utilities](#)

OpenSSH, [OpenSSH, Main Features](#)

- (see also SSH)
- client, [OpenSSH Clients](#)
 - scp, [Using the scp Utility](#)
 - sftp, [Using the sftp Utility](#)
 - ssh, [Using the ssh Utility](#)

- ECDSA keys
 - generating, [Generating Key Pairs](#)

- RSA keys
 - generating, [Generating Key Pairs](#)

- server, [Starting an OpenSSH Server](#)
 - starting, [Starting an OpenSSH Server](#)
 - stopping, [Starting an OpenSSH Server](#)

- ssh-add, [Configuring ssh-agent](#)
- ssh-agent, [Configuring ssh-agent](#)
- ssh-keygen

- ECDSA, [Generating Key Pairs](#)
- RSA, [Generating Key Pairs](#)

- using key-based authentication, [Using Key-based Authentication](#)

OpenSSL

- SSL (see SSL)
- TLS (see TLS)

ophelp, [Setting Events to Monitor](#)

opreport (see OProfile)

OProfile, [OProfile](#)

- /dev/oprofile/, [Understanding /dev/oprofile/](#)
- additional resources, [Additional Resources](#)
- configuring, [Configuring OProfile Using Legacy Mode](#)
 - separating profiles, [Separating Kernel and User-space Profiles](#)
- events
 - sampling rate, [Sampling Rate](#)
 - setting, [Setting Events to Monitor](#)
- Java, [OProfile Support for Java](#)
- monitoring the kernel, [Specifying the Kernel](#)
- opannotate, [Using opannotate](#)
- opcontrol, [Configuring OProfile Using Legacy Mode](#)
 - --no-vmlinux, [Specifying the Kernel](#)
 - --start, [Starting and Stopping OProfile Using Legacy Mode](#)
 - --vmlinux=, [Specifying the Kernel](#)
- ophelp, [Setting Events to Monitor](#)
- opreport, [Using opreport, Getting more detailed output on the modules](#)
 - on a single executable, [Using opreport on a Single Executable](#)
- oprofiled, [Starting and Stopping OProfile Using Legacy Mode](#)
 - log file, [Starting and Stopping OProfile Using Legacy Mode](#)
- overview of tools, [Overview of Tools](#)
- reading data, [Analyzing the Data](#)
- saving data, [Saving Data in Legacy Mode](#)
- starting, [Starting and Stopping OProfile Using Legacy Mode](#)
- SystemTap, [OProfile and SystemTap](#)
- unit mask, [Unit Masks](#)

oprofiled (see OProfile)

oprof_start, [Graphical Interface](#)

P

package

- kernel RPM, [Manually Upgrading the Kernel](#)

package groups

- listing package groups with Yum
- yum groups, [Listing Package Groups](#)

packages, [Working with Packages](#)

- dependencies, [Satisfying Unresolved Dependencies](#)
- determining file ownership with, [Practical and Common Examples of RPM Usage](#)
- displaying packages
 - yum info, [Displaying Package Information](#)
- displaying packages with Yum
 - yum info, [Displaying Package Information](#)
- downloading packages with Yum, [Downloading Packages](#)
- extra packages for Enterprise Linux (EPEL), [Finding RPM Packages](#)
- finding deleted files from, [Practical and Common Examples of RPM Usage](#)
- finding Red Hat RPM packages, [Finding RPM Packages](#)
- initial RPM repositories, [Finding RPM Packages](#)
- installing a package group with Yum, [Installing a Package Group](#)
- installing RPM, [Installing and Upgrading Packages](#)
- installing with Yum, [Installing Packages](#)
- kernel
 - for single,multicore and multiprocessor systems, [Overview of Kernel Packages](#)
- kernel-devel
 - kernel headers and makefiles, [Overview of Kernel Packages](#)
- kernel-headers
 - C header files files, [Overview of Kernel Packages](#)
- linux-firmware
 - firmware files, [Overview of Kernel Packages](#)
- listing packages with Yum
 - Glob expressions, [Searching Packages](#)
 - yum list available, [Listing Packages](#)
 - yum list installed, [Listing Packages](#)
 - yum repolist, [Listing Packages](#)
 - yum search, [Listing Packages](#)
- locating documentation for, [Practical and Common Examples of RPM Usage](#)
- obtaining list of files, [Practical and Common Examples of RPM Usage](#)
- perf
 - firmware files, [Overview of Kernel Packages](#)
- querying uninstalled, [Practical and Common Examples of RPM Usage](#)
- Red Hat Enterprise Linux installation media, [Finding RPM Packages](#)
- removing, [Uninstalling Packages](#)
- RPM, [RPM](#)
 - already installed, [Replacing Already-Installed Packages](#)
 - configuration file changes, [Preserving Changes in Configuration Files](#)
 - conflict, [Resolving File Conflicts](#)
 - failed dependencies, [Satisfying Unresolved Dependencies](#)
 - freshening, [Freshening Packages](#)
 - pristine sources, [RPM Design Goals](#)
 - querying, [Querying Packages](#)
 - removing, [Uninstalling Packages](#)
 - source and binary packages, [RPM](#)
 - tips, [Practical and Common Examples of RPM Usage](#)
 - uninstalling, [Uninstalling Packages](#)
 - verifying, [Verifying Packages](#)
- searching packages with Yum

- yum search, [Searching Packages](#)

- uninstalling packages with Yum, [Removing Packages](#)
- upgrading RPM, [Installing and Upgrading Packages](#)
- Yum instead of RPM, [RPM](#)

passwords

- shadow, [Shadow Passwords](#)

pdbedit program, Samba Distribution Programs

Postfix, Postfix

- default installation, [The Default Postfix Installation](#)

postfix, Mail Transport Agent (MTA) Configuration

Printer Configuration

- CUPS, [Printer Configuration](#)
- IPP Printers, [Adding an IPP Printer](#)
- LDP/LPR Printers, [Adding an LPD/LPR Host or Printer](#)
- Local Printers, [Adding a Local Printer](#)
- New Printer, [Starting Printer Setup](#)
- Print Jobs, [Managing Print Jobs](#)
- Samba Printers, [Adding a Samba \(SMB\) printer](#)
- Settings, [The Settings Page](#)
- Sharing Printers, [Sharing Printers](#)

printers (see Printer Configuration)

processes, Viewing System Processes

Procmail, Mail Delivery Agents

- additional resources, [Additional Resources](#)
- configuration, [Procmail Configuration](#)
- recipes, [Procmail Recipes](#)
 - delivering, [Delivering vs. Non-Delivering Recipes](#)
 - examples, [Recipe Examples](#)
 - flags, [Flags](#)
 - local lockfiles, [Specifying a Local Lockfile](#)
 - non-delivering, [Delivering vs. Non-Delivering Recipes](#)
 - SpamAssassin, [Spam Filters](#)
 - special actions, [Special Conditions and Actions](#)
 - special conditions, [Special Conditions and Actions](#)

ps, Using the ps Command

R

RAM, Viewing Memory Usage

rccp, Using the scp Utility

Red Hat Enterprise Linux installation media

- installable packages, [Finding RPM Packages](#)

rmmod, Unloading a Module

- (see also kernel module)

rpcclient program, Samba Distribution Programs

RPM, RPM

- additional resources, [Additional Resources](#)
- already installed, [Replacing Already-Installed Packages](#)
- basic modes, [Using RPM](#)
- checking package signatures, [Checking Package Signatures](#)
- configuration file changes, [Preserving Changes in Configuration Files](#)
 - conf.rpmsave, [Preserving Changes in Configuration Files](#)
- conflicts, [Resolving File Conflicts](#)
- dependencies, [Satisfying Unresolved Dependencies](#)
- design goals, [RPM Design Goals](#)
 - powerful querying, [RPM Design Goals](#)
 - system verification, [RPM Design Goals](#)
 - upgradability, [RPM Design Goals](#)
- determining file ownership with, [Practical and Common Examples of RPM Usage](#)
- documentation with, [Practical and Common Examples of RPM Usage](#)
- failed dependencies, [Satisfying Unresolved Dependencies](#)
- file conflicts
 - resolving, [Resolving File Conflicts](#)
- file name, [Installing and Upgrading Packages](#)
- finding and verifying RPM packages, [Finding and Verifying RPM Packages](#)
- finding deleted files with, [Practical and Common Examples of RPM Usage](#)
- finding Red Hat RPM packages, [Finding RPM Packages](#)
- freshening, [Freshening Packages](#)
- GnuPG, [Checking Package Signatures](#)
- installing, [Installing and Upgrading Packages](#)
- online documentation, [Additional Resources](#)
- querying, [Querying Packages](#)
- querying for file list, [Practical and Common Examples of RPM Usage](#)
- querying uninstalled packages, [Practical and Common Examples of RPM Usage](#)
- see also, [Additional Resources](#)
- tips, [Practical and Common Examples of RPM Usage](#)
- uninstalling, [Uninstalling Packages](#)
- upgrading, [Installing and Upgrading Packages](#)
- verification, [Verifying Packages](#)
- verifying, [Verifying Packages](#)
- website, [Additional Resources](#)

RPM Package Manager (see RPM)**RSA keys**

- generating, [Generating Key Pairs](#)

rsyslog, Viewing and Managing Log Files**S****Samba (see Samba)**

- Abilities, [Samba Features](#)
- Additional Resources, [Additional Resources](#)
 - installed documentation, [Installed Documentation](#)
 - useful websites, [Useful Websites](#)
- Browsing, [Samba Network Browsing](#)
- configuration, [Configuring a Samba Server](#), [Command Line Configuration](#)

- default, [Configuring a Samba Server](#)
- daemon, [Samba Daemons and Related Services](#)
 - nmbd, [Samba Daemons](#)
 - overview, [Samba Daemons](#)
 - smbd, [Samba Daemons](#)
 - winbindd, [Samba Daemons](#)
- encrypted passwords, [Encrypted Passwords](#)
- graphical configuration, [Graphical Configuration](#)
- Introduction, [Introduction to Samba](#)
- Network Browsing, [Samba Network Browsing](#)
 - Domain Browsing, [Domain Browsing](#)
 - WINS, [WINS \(Windows Internet Name Server\)](#)
- Programs, [Samba Distribution Programs](#)
 - net, [Samba Distribution Programs](#)
 - nmblookup, [Samba Distribution Programs](#)
 - pdbedit, [Samba Distribution Programs](#)
 - rpcclient, [Samba Distribution Programs](#)
 - smbcacls, [Samba Distribution Programs](#)
 - smbclient, [Samba Distribution Programs](#)
 - smbcontrol, [Samba Distribution Programs](#)
 - smbpasswd, [Samba Distribution Programs](#)
 - smbspool, [Samba Distribution Programs](#)
 - smbstatus, [Samba Distribution Programs](#)
 - smbtar, [Samba Distribution Programs](#)
 - testparm, [Samba Distribution Programs](#)
 - wbinfo, [Samba Distribution Programs](#)
- Reference, [Samba](#)
- Samba Printers, [Adding a Samba \(SMB\) printer](#)
- service
 - conditional restarting, [Starting and Stopping Samba](#)
 - reloading, [Starting and Stopping Samba](#)
 - restarting, [Starting and Stopping Samba](#)
 - starting, [Starting and Stopping Samba](#)
 - stopping, [Starting and Stopping Samba](#)
- share
 - connecting to via the command line, [Command Line](#)
 - connecting to with Nautilus, [Connecting to a Samba Share](#)
 - mounting, [Mounting the Share](#)
- smbclient, [Command Line](#)
- WINS, [WINS \(Windows Internet Name Server\)](#)
- with Windows NT 4.0, 2000, ME, and XP, [Encrypted Passwords](#)

scp (see OpenSSH)

security plug-in (see Security)

Security-Related Packages

- updating security-related packages, [Updating Packages](#)

Sendmail, Sendmail

- additional resources, [Additional Resources](#)
- aliases, [Masquerading](#)

- common configuration changes, [Common Sendmail Configuration Changes](#)
- default installation, [The Default Sendmail Installation](#)
- LDAP and, [Using Sendmail with LDAP](#)
- limitations, [Purpose and Limitations](#)
- masquerading, [Masquerading](#)
- purpose, [Purpose and Limitations](#)
- spam, [Stopping Spam](#)
- with UUCP, [Common Sendmail Configuration Changes](#)

sendmail, [Mail Transport Agent \(MTA\) Configuration](#)

sftp (see [OpenSSH](#))

shadow passwords

- overview of, [Shadow Passwords](#)

slapd (see [OpenLDAP](#))

smbcacls program, [Samba Distribution Programs](#)

smbclient, [Command Line](#)

smbclient program, [Samba Distribution Programs](#)

smbcontrol program, [Samba Distribution Programs](#)

smbpasswd program, [Samba Distribution Programs](#)

smbspool program, [Samba Distribution Programs](#)

smbstatus program, [Samba Distribution Programs](#)

smbtar program, [Samba Distribution Programs](#)

SpamAssassin

- using with Procmail, [Spam Filters](#)

ssh (see [OpenSSH](#))

SSH protocol

- authentication, [Authentication](#)
- configuration files, [Configuration Files](#)
 - system-wide configuration files, [Configuration Files](#)
 - user-specific configuration files, [Configuration Files](#)
- connection sequence, [Event Sequence of an SSH Connection](#)
- features, [Main Features](#)
- insecure protocols, [Requiring SSH for Remote Connections](#)
- layers
 - channels, [Channels](#)
 - transport layer, [Transport Layer](#)
- port forwarding, [Port Forwarding](#)
- requiring for remote login, [Requiring SSH for Remote Connections](#)
- security risks, [Why Use SSH?](#)
- version 1, [Protocol Versions](#)
- version 2, [Protocol Versions](#)
- X11 forwarding, [X11 Forwarding](#)

ssh-add, [Configuring ssh-agent](#)

ssh-agent, [Configuring ssh-agent](#)

SSL , [Setting Up an SSL Server](#)

- (see also Apache HTTP Server)

SSL server (see Apache HTTP Server)

stunnel, Securing Email Client Communications

system analysis

- OProfile (see OProfile)

system information

- cpu usage, [Viewing CPU Usage](#)
- file systems, [Viewing Block Devices and File Systems](#)
- gathering, [System Monitoring Tools](#)
- hardware, [Viewing Hardware Information](#)
- memory usage, [Viewing Memory Usage](#)
- processes, [Viewing System Processes](#)
 - currently running, [Using the top Command](#)

System Monitor, Using the System Monitor Tool, Using the System Monitor Tool, Using the System Monitor Tool, Using the System Monitor Tool

system-config-users (see user configuration and group configuration)

T

testparm program, Samba Distribution Programs

TLS , Setting Up an SSL Server

- (see also Apache HTTP Server)

top, Using the top Command

U

user configuration

- adding users, [Adding a New User](#)
- changing full name, [Modifying User Properties](#)
- changing home directory, [Modifying User Properties](#)
- changing login shell, [Modifying User Properties](#)
- changing password, [Modifying User Properties](#)
- command line configuration
 - passwd, [Adding a New User](#)
 - useradd, [Adding a New User](#)
- filtering list of users, [Viewing Users and Groups](#)
- modify groups for a user, [Modifying User Properties](#)
- modifying users, [Modifying User Properties](#)
- viewing list of users, [Using the User Manager Tool](#)

User Manager (see user configuration)

user private groups (see groups)

- and shared directories, [Creating Group Directories](#)

useradd command

- user account creation using, [Adding a New User](#)

users (see user configuration)

- introducing, [Managing Users and Groups](#)

- tools for management of
 - User Manager, [Using Command Line Tools](#)
 - useradd, [Using Command Line Tools](#)
- UID, [Managing Users and Groups](#)

V

virtual host (see Apache HTTP Server)

vsftpd

- additional resources, [Additional Resources](#)
 - installed documentation, [Installed Documentation](#)
 - online documentation, [Online Documentation](#)
- encrypting, [Encrypting vsftpd Connections Using SSL](#)
- multihome configuration, [Starting Multiple Copies of vsftpd](#)
- restarting, [Starting and Stopping vsftpd](#)
- securing, [Encrypting vsftpd Connections Using SSL](#), [SELinux Policy for vsftpd](#)
- SELinux, [SELinux Policy for vsftpd](#)
- SSL, [Encrypting vsftpd Connections Using SSL](#)
- starting, [Starting and Stopping vsftpd](#)
- starting multiple copies of, [Starting Multiple Copies of vsftpd](#)
- status, [Starting and Stopping vsftpd](#)
- stopping, [Starting and Stopping vsftpd](#)

W

wbinfo program, [Samba Distribution Programs](#)

web server (see Apache HTTP Server)

window managers (see X)

Windows 2000

- connecting to shares using Samba, [Encrypted Passwords](#)

Windows 98

- connecting to shares using Samba, [Encrypted Passwords](#)

Windows ME

- connecting to shares using Samba, [Encrypted Passwords](#)

Windows NT 4.0

- connecting to shares using Samba, [Encrypted Passwords](#)

Windows XP

- connecting to shares using Samba, [Encrypted Passwords](#)

X

X

- /etc/X11/xorg.conf
 - Boolean values for, [The Structure of the Configuration](#)
 - Device, [The Device section](#)
 - DRI, [The DRI section](#)
 - Files section, [The Files section](#)
 - InputDevice section, [The InputDevice section](#)
 - introducing, [The xorg.conf.d Directory](#), [The xorg.conf File](#)

- Monitor, [The Monitor section](#)
- Screen, [The Screen section](#)
- Section tag, [The Structure of the Configuration](#)
- ServerFlags section, [The ServerFlags section](#)
- ServerLayout section, [The ServerLayout Section](#)
- structure of, [The Structure of the Configuration](#)

- additional resources, [Additional Resources](#)
 - installed documentation, [Installed Documentation](#)
 - useful websites, [Useful Websites](#)

- configuration directory
 - /etc/X11/xorg.conf.d, [The xorg.conf.d Directory](#)

- configuration files
 - /etc/X11/ directory, [X Server Configuration Files](#)
 - /etc/X11/xorg.conf, [The xorg.conf File](#)
 - options within, [X Server Configuration Files](#)
 - server options, [The xorg.conf.d Directory](#), [The xorg.conf File](#)

- desktop environments
 - GNOME, [Desktop Environments](#)
 - KDE, [Desktop Environments](#)

- fonts
 - Fontconfig, [Fonts](#)
 - Fontconfig, adding fonts to, [Adding Fonts to Fontconfig](#)
 - FreeType, [Fonts](#)
 - introducing, [Fonts](#)
 - Xft, [Fonts](#)

- introducing, [The X Window System](#)
- runlevels and, [Runlevels, targets, and X](#)
- window managers
 - kwin, [Window Managers](#)
 - metacity, [Window Managers](#)
 - mwm, [Window Managers](#)

- X clients, [The X Window System, Desktop Environments and Window Managers](#)
 - desktop environments, [Desktop Environments](#)
 - window managers, [Window Managers](#)

- X server, [The X Window System](#)
 - features of, [The X Server](#)

X Window System (see X)

X.500 (see OpenLDAP)

X.500 Lite (see OpenLDAP)

Xorg (see Xorg)

Y

Yum

- configuring plug-ins, [Enabling, Configuring, and Disabling Yum Plug-ins](#)
- configuring Yum and Yum repositories, [Configuring Yum and Yum Repositories](#)
- disabling plug-ins, [Enabling, Configuring, and Disabling Yum Plug-ins](#)
- displaying packages

- yum info, [Displaying Package Information](#)
- displaying packages with Yum
 - yum info, [Displaying Package Information](#)
- downloading packages with Yum, [Downloading Packages](#)
- enabling plug-ins, [Enabling, Configuring, and Disabling Yum Plug-ins](#)
- installing a package group with Yum, [Installing a Package Group](#)
- installing with Yum, [Installing Packages](#)
- listing package groups with Yum
 - yum groups list, [Listing Package Groups](#)
- listing packages with Yum
 - Glob expressions, [Searching Packages](#)
 - yum list, [Listing Packages](#)
 - yum list available, [Listing Packages](#)
 - yum list installed, [Listing Packages](#)
 - yum repolist, [Listing Packages](#)
- packages, [Working with Packages](#)
- plug-ins
 - aliases, [Working with Yum Plug-ins](#)
 - kabi, [Working with Yum Plug-ins](#)
 - langpacks, [Working with Yum Plug-ins](#)
 - product-id, [Working with Yum Plug-ins](#)
 - yum-changelog, [Working with Yum Plug-ins](#)
 - yum-tmprepo, [Working with Yum Plug-ins](#)
 - yum-verify, [Working with Yum Plug-ins](#)
 - yum-versionlock, [Working with Yum Plug-ins](#)
- repository, [Adding, Enabling, and Disabling a Yum Repository](#), [Creating a Yum Repository](#)
- searching packages with Yum
 - yum search, [Searching Packages](#)
- setting [main] options, [Setting \[main\] Options](#)
- setting [repository] options, [Setting \[repository\] Options](#)
- uninstalling packages with Yum, [Removing Packages](#)
- variables, [Using Yum Variables](#)
- Yum plug-ins, [Yum Plug-ins](#)
- Yum repositories
 - configuring Yum and Yum repositories, [Configuring Yum and Yum Repositories](#)

Yum Updates

- checking for updates, [Checking For Updates](#)
- updating a single package, [Updating Packages](#)
- updating all packages and dependencies, [Updating Packages](#)
- updating packages, [Updating Packages](#)
- updating security-related packages, [Updating Packages](#)