

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.

### 19.3.3.6. 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.

### 19.3.3.7. 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.

## 19.3.4. Mail Transport Agent (MTA) Configuration

A *Mail Transport Agent* (MTA) is essential for sending email. A *Mail User Agent* (MUA) such as **Evolution**, **Thunderbird**, 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 **/bin/mail** command to send email containing log messages to the **root** user of the local system.

Red Hat Enterprise Linux 6 provides two MTAs: Postfix and Sendmail. If both are installed, Postfix is the default MTA.

## 19.4. 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.

### 19.4.1. 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 (`#`). Note that it uses paths relative to the current directory.

- **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.

## 19.4.2. Procmail Recipes

New users often find the construction of recipes the most difficult part of learning to use Procmail. 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 Procmail recipes are written, regardless of regular expressions, makes it easy to learn by example. To see example Procmail recipes, see [Section 19.4.2.5, “Recipe Examples”](#).

Procmail recipes take the following form:

```
:0 [flags] [: lockfile-name ]
* [ condition_1_special-condition-character condition_1_regular_expression
]
* [ condition_2_special-condition-character condition-2_regular_expression
]
* [ condition_N_special-condition-character condition-N_regular_expression
]
    special-action-character
    action-to-perform
```

The first two characters in a Procmail recipe are a colon and a zero. Various flags can be placed after the zero to control how Procmail 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 19.4.2.4, “Special Conditions and Actions”](#) for more information.

### 19.4.2.1. 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, Procmail 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.

### 19.4.2.2. Flags

Flags are essential to determine how or if a recipe's conditions are compared to a message. The **egrep** utility is used internally for matching of the conditions. 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.

### 19.4.2.3. 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.

### 19.4.2.4. 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.

### 19.4.2.5. 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(1)** 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.

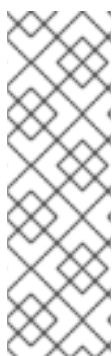
Consult the many Procmal online resources available in [Section 19.6, “Additional Resources”](#) for more detailed and powerful recipes.

#### 19.4.2.6. Spam Filters

Because it is called by Sendmail, Postfix, and Fetchmail upon receiving new emails, Procmal can be used as a powerful tool for combating spam.

This is particularly true when Procmal 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 8.2.4, “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:

```
~]# service spamassassin start
```

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 12, Services and Daemons](#) 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
```

## 19.5. 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.

### 19.5.1. 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 user names, 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 user names and passwords as they are passed over the network.

### 19.5.1.1. 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.

### 19.5.1.2. 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 `service dovecot restart` command to restart the **dovecot** daemon.

Alternatively, the **stunnel** command can be used as an 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.

See [Using stunnel](#) in the Red Hat Enterprise Linux 6 Security Guide for instructions on how to install **stunnel** and create its basic configuration. To configure **stunnel** as a wrapper for **IMAPS** and **POP3S**, add the following lines to the `/etc/stunnel/stunnel.conf` configuration file:

```
[pop3s]
accept = 995
connect = 110

[imaps]
accept = 993
connect = 143
```

The Security Guide also explains how to start and stop **stunnel**. Once you start it, it is possible to use an **IMAP** or a **POP** email client and connect to the email server using SSL encryption.

## 19.6. ADDITIONAL RESOURCES

The following is a list of additional documentation about email applications.

### 19.6.1. 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 on how 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.

### 19.6.2. Online Documentation

- [How to configure postfix with TLS?](#) — A Red Hat Knowledgebase article that describes configuring postfix to use TLS.
- The Red Hat Knowledgebase article [How to Configure a System to Manage Multiple Virtual Mailboxes Using Postfix and Dovecot](#) describes managing multiple virtual users under one real-user account using Postfix as Mail Transporting Agent (MTA) and Dovecot as IMAP server.
- <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://www.fetchmail.info/fetchmail-FAQ.html> — A thorough FAQ about Fetchmail.
- <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.spamassassin.org/> — The official site of the SpamAssassin project.

### 19.6.3. Related Books

- *Sendmail Milners: 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 20. DIRECTORY SERVERS

### 20.1. 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”.

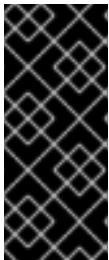
Like X.500, LDAP organizes information in a hierarchical manner using directories. These directories can store a variety of information such as names, addresses, or phone numbers, and can even be used in a manner similar to the *Network Information Service* (NIS), enabling anyone to access their account 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.

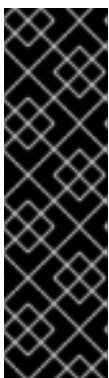
#### 20.1.1. Introduction to LDAP

Using a client-server architecture, LDAP provides a 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 *Transport Layer Security* (TLS) cryptographic protocol can be used to prevent an attacker from intercepting the transmission.



#### IMPORTANT

The OpenLDAP suite in Red Hat Enterprise Linux 6 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](#).



#### IMPORTANT

Due to the vulnerability described in [Resolution for POODLE SSLv3.0 vulnerability \(CVE-2014-3566\) for components that do not allow SSLv3 to be disabled via configuration settings](#), Red Hat recommends that you do not rely on the **SSLv3** protocol for security. OpenLDAP is one of the system components that do not provide configuration parameters that allow **SSLv3** to be effectively disabled. To mitigate the risk, it is recommended that you use the **stunnel** command to provide a secure tunnel, and disable **stunnel** from using **SSLv3**. For more information on using **stunnel**, see the [Red Hat Enterprise Linux 6 Security Guide](#).

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.

### 20.1.1.1. 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.

### 20.1.1.2. 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), and Transport Layer Security (TLS) 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.

### 20.1.1.3. 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 20.1.2, “Installing the OpenLDAP Suite”](#) for more information on required packages.
2. Customize the configuration as described in [Section 20.1.3, “Configuring an OpenLDAP Server”](#).
3. Start the **slapd** service as described in [Section 20.1.4, “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.

### 20.1.2. Installing the OpenLDAP Suite

The suite of OpenLDAP libraries and tools is provided by the following packages:

**Table 20.1. List of OpenLDAP packages**

Package	Description
openldap	A package containing the libraries necessary to run the OpenLDAP server and client applications.
openldap-clients	A package containing the command-line utilities for viewing and modifying directories on an LDAP server.
openldap-servers	A package containing both the services and utilities to configure and run an LDAP server. This includes the <i>Standalone LDAP Daemon</i> , <b>slapd</b> .
compat-openldap	A package containing the OpenLDAP compatibility libraries.

Additionally, the following packages are commonly used along with the LDAP server:

**Table 20.2. List of commonly installed additional LDAP packages**

Package	Description
---------	-------------

Package	Description
sssd	A package containing <b>the System Security Services Daemon (SSSD)</b> , a set of daemons to manage access to remote directories and authentication mechanisms. It provides the Name Service Switch (NSS) and the Pluggable Authentication Modules (PAM) interfaces toward the system and a pluggable back-end system to connect to multiple different account sources.
mod_authz_ldap	A package containing <b>mod_authz_ldap</b> , the LDAP authorization module for the Apache HTTP Server. This module uses the short form of the distinguished name for a subject and the issuer of the client SSL certificate to determine the distinguished name of the user within an LDAP directory. It is also capable of authorizing users based on attributes of that user's LDAP directory entry, determining access to assets based on the user and group privileges of the asset, and denying access for users with expired passwords. Note that the <b>mod_ssl</b> module is required when using the <b>mod_authz_ldap</b> module.

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 8.2.4, “Installing Packages”](#).

### 20.1.2.1. Overview of OpenLDAP Server Utilities

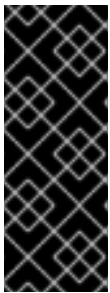
To perform administrative tasks, the openldap-servers package installs the following utilities along with the **slapd** service:

**Table 20.3. List of OpenLDAP server utilities**

Command	Description
<b>slapacl</b>	Allows you to check the access to a list of attributes.
<b>slapadd</b>	Allows you to add entries from an LDIF file to an LDAP directory.
<b>slapauth</b>	Allows you to check a list of IDs for authentication and authorization permissions.
<b>slapcat</b>	Allows you to pull entries from an LDAP directory in the default format and save them in an LDIF file.

Command	Description
<b>slapdn</b>	Allows you to check a list of Distinguished Names (DNs) based on available schema syntax.
<b>slapindex</b>	Allows you to re-index the <b>slapd</b> directory based on the current content. Run this utility whenever you change indexing options in the configuration file.
<b>slappasswd</b>	Allows you to create an encrypted user password to be used with the <b>ldapmodify</b> utility, or in the <b>slapd</b> configuration file.
<b>slapschema</b>	Allows you to check the compliance of a database with the corresponding schema.
<b>slaptest</b>	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 20.1.6.1, “Installed Documentation”](#).



### IMPORTANT

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
```



### WARNING

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:

```
~]# service slapd stop
Stopping slapd:
[ OK ]
```

For more information on how to start, stop, restart, and check the current status of the **slapd** service, see [Section 20.1.4, “Running an OpenLDAP Server”](#).

### 20.1.2.2. 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 20.4. List of OpenLDAP client utilities**

Command	Description
<b>ldapadd</b>	Allows you to add entries to an LDAP directory, either from a file, or from standard input. It is a symbolic link to <b>ldapmodify -a</b> .
<b>ldapcompare</b>	Allows you to compare given attribute with an LDAP directory entry.
<b>ldapdelete</b>	Allows you to delete entries from an LDAP directory.
<b>ldapexop</b>	Allows you to perform extended LDAP operations.
<b>ldapmodify</b>	Allows you to modify entries in an LDAP directory, either from a file, or from standard input.
<b>ldapmodrdn</b>	Allows you to modify the RDN value of an LDAP directory entry.
<b>ldappasswd</b>	Allows you to set or change the password for an LDAP user.
<b>ldapsearch</b>	Allows you to search LDAP directory entries.
<b>ldapurl</b>	Allows you to compose or decompose LDAP URLs.
<b>ldapwhoami</b>	Allows you to perform a <b>whoami</b> 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.

### 20.1.2.3. 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**.

### 20.1.3. Configuring an OpenLDAP Server

By default, OpenLDAP stores its configuration in the `/etc/openldap/` directory. [Table 20.5, “List of OpenLDAP configuration files and directories”](#) highlights the most important files and directories within this directory.

**Table 20.5. List of OpenLDAP configuration files and directories**

Path	Description
<code>/etc/openldap/ldap.conf</code>	The configuration file for client applications that use the OpenLDAP libraries. This includes <b>ldapadd</b> , <b>ldapsearch</b> , <b>Evolution</b> , etc.

Path	Description
<code>/etc/openldap/slapd.d/</code>	The directory containing the <b>slapd</b> configuration.

In Red Hat Enterprise Linux 6, the **slapd** service uses a configuration database located in the `/etc/openldap/slapd.d/` directory and only reads the old `/etc/openldap/slapd.conf` configuration file if this directory does not exist. If you have an existing **slapd.conf** file from a previous installation, you can either wait for the `openldap-servers` package to convert it to the new format the next time you update this package, or type the following at a shell prompt as **root** to convert it immediately:

```
~]# 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 20.1.2.1, “Overview of OpenLDAP Server Utilities”](#).



### IMPORTANT

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/` directory directly.

#### 20.1.3.1. 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 20.6, “Available olcAllows options”](#). The default option is **bind\_v2**.

**Table 20.6. Available olcAllows options**

Option	Description
<b>bind_v2</b>	Enables the acceptance of LDAP version 2 bind requests.
<b>bind_anon_cred</b>	Enables an anonymous bind when the Distinguished Name (DN) is empty.
<b>bind_anon_dn</b>	Enables an anonymous bind when the Distinguished Name (DN) is <i>not</i> empty.
<b>update_anon</b>	Enables processing of anonymous update operations.
<b>proxy_authz_anon</b>	Enables processing of anonymous proxy authorization control.

**Example 20.1. 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 20.2. 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 20.3. 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 20.7, “Available `olcDisallows` options”](#). No features are disabled by default.

**Table 20.7. Available `olcDisallows` options**

Option	Description
<b><code>bind_anon</code></b>	Disables the acceptance of anonymous bind requests.
<b><code>bind_simple</code></b>	Disables the simple bind authentication mechanism.

Option	Description
<b>tls_2_anon</b>	Disables the enforcing of an anonymous session when the STARTTLS command is received.
<b>tls_authc</b>	Disallows the STARTTLS command when authenticated.

#### Example 20.4. 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 20.5. 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 20.6. 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 20.7. 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 20.8. Using the olcWriteTimeout directive**

```
olcWriteTimeout: 180
```

### **20.1.3.2. 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={2}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 20.9. 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,dc=my-domain,dc=com**.

#### **Example 20.10. Using the olcRootDN directive**

```
olcRootDN: cn=root,dc=example,dc=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:

```
~]$ slappaswd
New password:
Re-enter new password:
{SSHA}WczWsyPEnMchFf1GRTweq2q7XJcvmSxD
```

**Example 20.11. 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 20.12. Using the olcSuffix directive**

```
olcSuffix: dc=example,dc=com
```

**20.1.3.3. Extending Schema**

Since OpenLDAP 2.3, the `/etc/openldap/slapd.d/cn=config/cn=schema/` 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>.

**20.1.4. 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 12, Services and Daemons](#).

**20.1.4.1. Starting the Service**

To run the **slapd** service, type the following at a shell prompt:

```
~]# service slapd start
Starting slapd: [ OK ]
```

If you want the service to start automatically at the boot time, use the following command:

```
~]# chkconfig slapd on
```

Note that you can also use the **Service Configuration** utility as described in [Section 12.2.1.1, “Enabling and Disabling a Service”](#).

#### 20.1.4.2. Stopping the Service

To stop the running **slapd** service, type the following at a shell prompt:

```
~]# service slapd stop
Stopping slapd: [ OK ]
```

To prevent the service from starting automatically at the boot time, type:

```
~]# chkconfig slapd off
```

Alternatively, you can use the **Service Configuration** utility as described in [Section 12.2.1.1, “Enabling and Disabling a Service”](#).

#### 20.1.4.3. Restarting the Service

To restart the running **slapd** service, type the following at a shell prompt:

```
~]# service slapd restart
Stopping slapd: [ OK ]
Starting slapd: [ OK ]
```

This stops the service, and then starts it again. Use this command to reload the configuration.

#### 20.1.4.4. Checking the Service Status

To check whether the service is running, type the following at a shell prompt:

```
~]# service slapd status
slapd (pid 3672) is running...
```

#### 20.1.5. 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 20.1.2, “Installing the OpenLDAP Suite”](#) and [Section 20.1.3, “Configuring an OpenLDAP Server”](#). On a client, type the following at a shell prompt:

```
~]# yum install openldap openldap-clients sssd
```

Chapter 13, *Configuring Authentication* provides detailed instructions on how to configure applications to use LDAP for authentication.

### 20.1.5.1. 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 20.8, “Commonly used LDAP migration scripts”](#).

**Table 20.8. Commonly used LDAP migration scripts**

Existing Name Service	Is LDAP Running?	Script to Use
<code>/etc</code> flat files	yes	<code>migrate_all_online.sh</code>
<code>/etc</code> 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.

### 20.1.6. 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*.

### 20.1.6.1. 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.

### 20.1.6.2. Useful Websites

<http://www.openldap.org/doc/admin24/>

The current version of the *OpenLDAP Software Administrator's Guide*.

### 20.1.6.3. 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 21. FILE AND PRINT SERVERS

### 21.1. SAMBA

**Samba** is the standard open source Windows interoperability suite of programs for Linux. It implements 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 the following command as **root**:

```
~]# yum install samba
```

For more information on installing packages with Yum, see [Section 8.2.4, “Installing Packages”](#).

#### 21.1.1. Introduction to Samba

Samba is an important component to seamlessly integrate Linux Servers and Desktops into Active Directory (AD) environments. It can function both as a domain controller (NT4-style) or as a regular domain member (AD or NT4-style).

##### What Samba can do:

- Serve directory trees and printers to Linux, UNIX, and Windows clients
- Assist in network browsing (with 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

##### What Samba cannot do:

- Act as a BDC for a Windows PDC (and vice versa)
- Act as an Active Directory domain controller

#### 21.1.2. Samba Daemons and Related Services

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 and Active Directory 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 **winbind** daemon is controlled by the **winbind** service and does not require the **smb** service to be started in order to operate. **winbind** 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.

For information on how to configure **winbind** for authentication, see [Section 13.1.2.3, “Configuring Winbind Authentication”](#).



#### **NOTE**

See [Section 21.1.11, “Samba Distribution Programs”](#) for a list of utilities included in the Samba distribution.

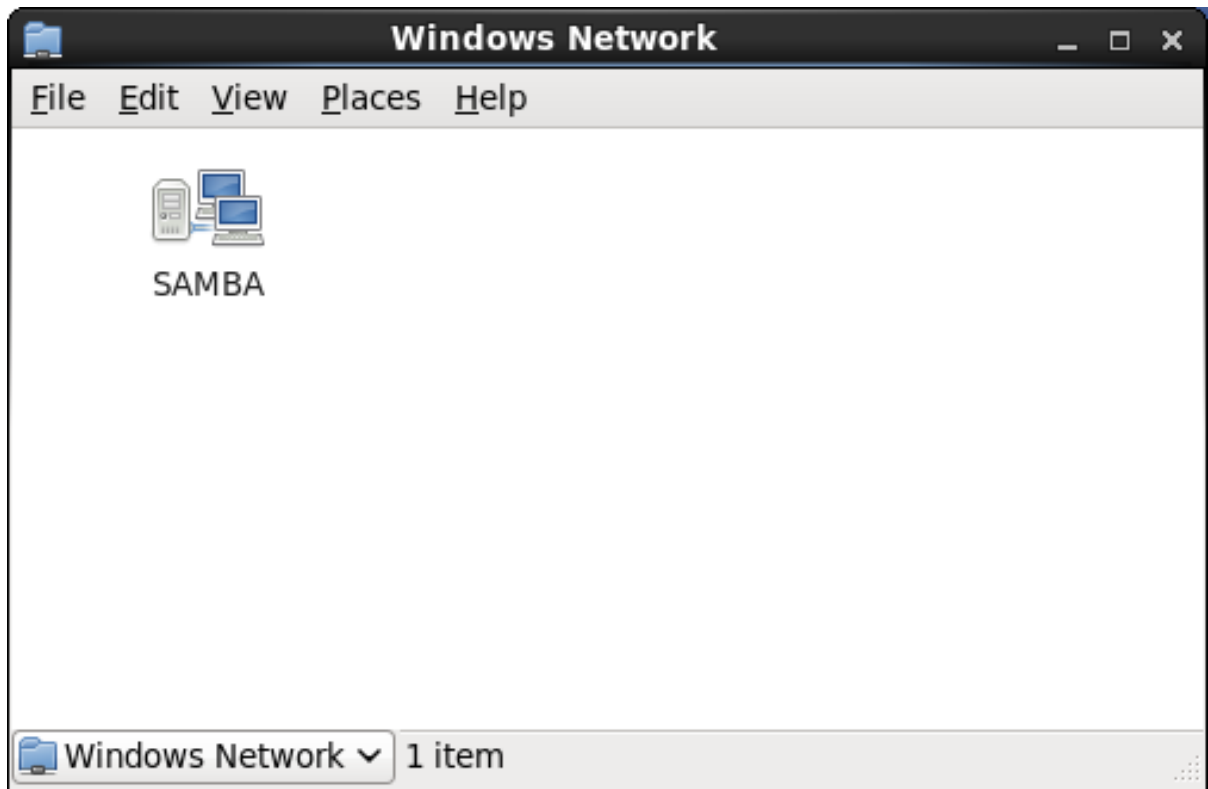
### **21.1.3. Connecting to a Samba Share**

You can use either **Nautilus** or command line to connect to available Samba shares.

#### **Procedure 21.1. Connecting to a Samba Share Using Nautilus**

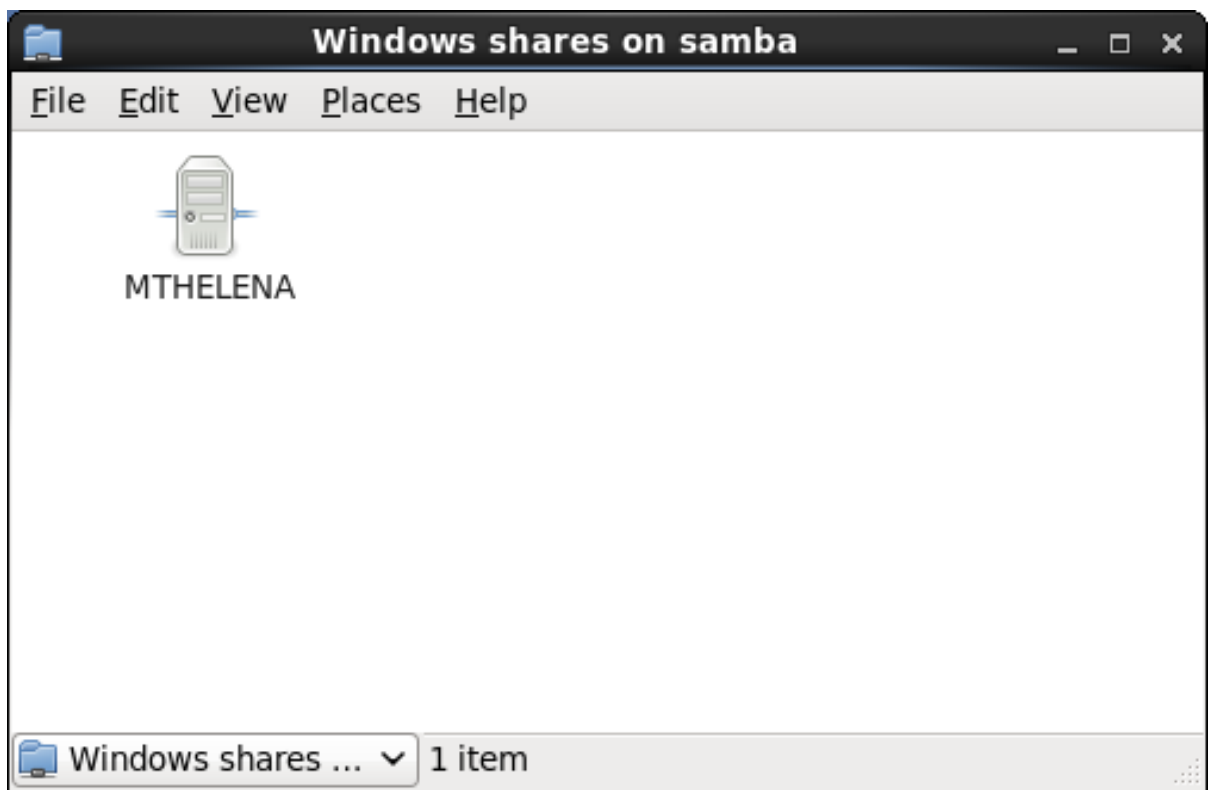
1. To view a list of Samba workgroups and domains on your network, select **Places** → **Network** from the GNOME panel, and then select the desired network. Alternatively, type **smb :** in the **File** → **Open Location** bar of **Nautilus**.

As shown in [Figure 21.1, “SMB Workgroups in Nautilus”](#), an icon appears for each available **SMB** workgroup or domain on the network.



**Figure 21.1. SMB Workgroups in Nautilus**

2. Double-click one of the workgroup or domain icon to view a list of computers within the workgroup or domain.



**Figure 21.2. SMB Machines in Nautilus**

3. As displayed in [Figure 21.2, "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
```

### Procedure 21.2. Connecting to a Samba Share Using the Command Line

1. To query the network for Samba servers, use the **findsmb** command. For each server found, it displays its **IP** address, NetBIOS name, workgroup name, operating system, and **SMB** server version:

```
findsmb
```

2. To connect to a Samba share from a shell prompt, type the following command:

```
smbclient //hostname/sharename -U username
```

Replace *hostname* with the host name 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 user name 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 want to browse the contents of your home directory, replace *sharename* with your user name. If the **-U** switch is not used, the user name of the current user is passed to the Samba server.

3. To exit **smbclient**, type **exit** at the **smb:\>** prompt.

#### 21.1.3.1. 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/*.

For more information about mounting a samba share, see the `mount.cifs(8)` manual page.



## 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 the following command as **root**:

```
~]# yum install cifs-utils
```

For more information on installing packages with Yum, see [Section 8.2.4, “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 the **cifs.upcall(8)** manual page.



## WARNING

Some CIFS servers require plain text passwords for authentication. Support for plain text password authentication can be enabled using the following command as **root**:

```
~]# echo 0x37 > /proc/fs/cifs/SecurityFlags
```

WARNING: This operation can expose passwords by removing password encryption.

## 21.1.4. 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.

### 21.1.4.1. 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/>.

### 21.1.4.2. 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**:

```
~]# service smb restart
```

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):

### Example 21.1. An Example Configuration of a Samba Server

```
[sharename]
comment = Insert a comment here
path = /home/share/
valid users = tfox carole
writable = yes
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.

#### 21.1.4.3. Encrypted Passwords

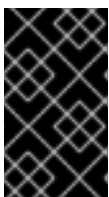
Encrypted passwords are enabled by default because it is more secure to use them. To create a user with an encrypted password, use the **smbpasswd** utility:

```
smbpasswd -a username
```

#### 21.1.5. Starting and Stopping Samba

To start a Samba server, type the following command in a shell prompt, as **root**:

```
~]# service smb start
```



#### 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**:

```
~]# service smb stop
```

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**:

```
~]# service smb restart
```



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**:

```
~]# service smb condrestart
```

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**:

```
~]# service smb reload
```

By default, the **smb** service does *not* start automatically at boot time. To configure Samba to start at boot time, use an initscript utility, such as `/sbin/chkconfig`, `/usr/sbin/ntsysv`, or the **Services Configuration Tool** program. See [Chapter 12, Services and Daemons](#) for more information regarding these tools.

## 21.1.6. Samba Server Types and the `smb.conf` File

Samba configuration is straightforward. All modifications to Samba are done in the `/etc/samba/smb.conf` configuration file. Although the default `smb.conf` file is well documented, it does not address complex topics such as LDAP, Active Directory, and the numerous domain controller implementations.

The following sections describe the different ways a Samba server can be configured. Keep in mind your needs and the changes required to the `/etc/samba/smb.conf` file for a successful configuration.

### 21.1.6.1. Stand-alone Server

A stand-alone server can be a workgroup server or a member of a workgroup environment. A stand-alone server is not a domain controller and does not participate in a domain in any way. The following examples include several user-level security configurations. For more information on security modes, see [Section 21.1.7, “Samba Security Modes”](#).

#### Anonymous Read-Only

The following `/etc/samba/smb.conf` file shows a sample configuration needed to implement anonymous read-only file sharing. Two directives are used to configure anonymous access – `map to guest = Bad user` and `guest account = nobody`.

#### Example 21.2. An Example Configuration of a Anonymous Read-Only Samba Server

```
[global]
workgroup = DOCS
netbios name = DOCS_SRV
security = user
guest account = nobody # default value
```

```
map to guest = Bad user
```

```
[data]
comment = Documentation Samba Server
path = /export
read only = yes
guest ok = yes
```

### Anonymous Read/Write

The following `/etc/samba/smb.conf` file shows a sample configuration needed to implement anonymous read/write file sharing. To enable anonymous read/write file sharing, set the ***read only*** directive to **no**. The ***force user*** and ***force group*** directives are also added to enforce the ownership of any newly placed files specified in the share.



#### NOTE

Although having an anonymous read/write server is possible, it is not recommended. Any files placed in the share space, regardless of user, are assigned the user/group combination as specified by a generic user (***force user***) and group (***force group***) in the `/etc/samba/smb.conf` file.

### Example 21.3. An Example Configuration of a Anonymous Read/Write Samba Server

```
[global]
workgroup = DOCS
security = user
guest account = nobody # default value
map to guest = Bad user

[data]
comment = Data
path = /export
guest ok = yes
writeable = yes
force user = user
force group = group
```

### Anonymous Print Server

The following `/etc/samba/smb.conf` file shows a sample configuration needed to implement an anonymous print server. Setting ***browseable*** to **no** as shown does not list the printer in Windows **Network Neighborhood**. Although hidden from browsing, configuring the printer explicitly is possible. By connecting to **DOCS\_SRV** using NetBIOS, the client can have access to the printer if the client is also part of the **DOCS** workgroup. It is also assumed that the client has the correct local printer driver installed, as the ***use client driver*** directive is set to **yes**. In this case, the Samba server has no responsibility for sharing printer drivers to the client.

### Example 21.4. An Example Configuration of a Anonymous Print Samba Server

```
[global]
workgroup = DOCS
```

```
netbios name = DOCS_SRV
security = user
map to guest = Bad user
printing = cups
```

```
[printers]
comment = All Printers
path = /var/spool/samba
guest ok = yes
printable = yes
use client driver = yes
browseable = yes
```

### Secure Read/Write File and Print Server

The following `/etc/samba/smb.conf` file shows a sample configuration needed to implement a secure read/write file and print server. Setting the *security* directive to **user** forces Samba to authenticate client connections. Notice the **[homes]** share does not have a *force user* or *force group* directive as the **[public]** share does. The **[homes]** share uses the authenticated user details for any files created as opposed to the *force user* and *force group* in **[public]**.

#### Example 21.5. An Example Configuration of a Secure Read/Write File and Print Samba Server

```
[global]
workgroup = DOCS
netbios name = DOCS_SRV
security = user
printcap name = cups
disable spools = yes
show add printer wizard = no
printing = cups

[homes]
comment = Home Directories
valid users = %S
read only = no
browseable = no

[public]
comment = Data
path = /export
force user = docsbot
force group = users
guest ok = yes

[printers]
comment = All Printers
path = /var/spool/samba
printer admin = john, ed, @admins
create mask = 0600
guest ok = yes
printable = yes
use client driver = yes
browseable = yes
```

### 21.1.6.2. Domain Member Server

A domain member, while similar to a stand-alone server, is logged into a domain controller (either Windows or Samba) and is subject to the domain's security rules. An example of a domain member server would be a departmental server running Samba that has a machine account on the Primary Domain Controller (PDC). All of the department's clients still authenticate with the PDC, and desktop profiles and all network policy files are included. The difference is that the departmental server has the ability to control printer and network shares.

#### Active Directory Domain Member Server

To implement an Active Directory domain member server, follow procedure below:

#### Procedure 21.3. Adding a Member Server to an Active Directory Domain

1. Create the `/etc/samba/smb.conf` configuration file on a member server to be added to the Active Directory domain. Add the following lines to the configuration file:

```
[global]
realm = EXAMPLE.COM
security = ADS
encrypt passwords = yes
# Optional. Use only if Samba cannot determine the Kerberos server
automatically.
password server = kerberos.example.com
```

With the above configuration, Samba authenticates users for services being run locally but is also a client of the Active Directory. Ensure that your kerberos *realm* parameter is shown in all caps (for example `realm = EXAMPLE.COM`). Since Windows 2000/2003/2008 requires Kerberos for Active Directory authentication, the *realm* directive is required. If Active Directory and Kerberos are running on different servers, the *password server* directive is required to help the distinction.

2. Configure Kerberos on the member server. Create the `/etc/krb5.conf` configuration file with the following content:

```
[logging]
default = FILE:/var/log/krb5libs.log

[libdefaults]
default_realm = AD.EXAMPLE.COM
dns_lookup_realm = true
dns_lookup_kdc = true
ticket_lifetime = 24h
renew_lifetime = 7d
rdns = false
forwardable = false

[realms]
# Define only if DNS lookups are not working
# AD.EXAMPLE.COM = {
#   kdc = server.ad.example.com
#   admin_server = server.ad.example.com
#   master_kdc = server.ad.example.com
```

```
# }

[domain_realm]
# Define only if DNS lookups are not working
# .ad.example.com = AD.EXAMPLE.COM
# ad.example.com = AD.EXAMPLE.COM
```

Uncomment the `[realms]` and `[domain_realm]` sections if DNS lookups are not working.

For more information on Kerberos, and the `/etc/krb5.conf` file, see the *Using Kerberos* section of the [Red Hat Enterprise Linux 6 Managing Single Sign-On and Smart Cards](#).

- To join an Active Directory server, type the following command as **root** on the member server:

```
~]# net ads join -U administrator%password
```

The **net** command authenticates as **Administrator** using the NT LAN Manager (NTLM) protocol and creates the machine account. Then **net** uses the machine account credentials to authenticate with Kerberos.



#### NOTE

Since **security = ads** and not **security = user** is used, a local password back end such as **smbpasswd** is not needed. Older clients that do not support **security = ads** are authenticated as if **security = domain** had been set. This change does not affect functionality and allows local users not previously in the domain.

### Windows NT4-based Domain Member Server

The following `/etc/samba/smb.conf` file shows a sample configuration needed to implement a Windows NT4-based domain member server. Becoming a member server of an NT4-based domain is similar to connecting to an Active Directory. The main difference is NT4-based domains do not use Kerberos in their authentication method, making the `/etc/samba/smb.conf` file simpler. In this instance, the Samba member server functions as a pass through to the NT4-based domain server.

#### Example 21.6. An Example Configuration of Samba Windows NT4-based Domain Member Server

```
[global]
workgroup = DOCS
netbios name = DOCS_SRV
security = domain

[homes]
comment = Home Directories
valid users = %S
read only = no
browseable = no

[public]
comment = Data
path = /export
```

```
force user = docsbot
force group = users
guest ok = yes
```

Having Samba as a domain member server can be useful in many situations. There are times where the Samba server can have other uses besides file and printer sharing. It may be beneficial to make Samba a domain member server in instances where Linux-only applications are required for use in the domain environment. Administrators appreciate keeping track of all machines in the domain, even if not Windows-based. In the event the Windows-based server hardware is deprecated, it is quite easy to modify the `/etc/samba/smb.conf` file to convert the server to a Samba-based PDC. If Windows NT-based servers are upgraded to Windows 2000/2003/2008 the `/etc/samba/smb.conf` file is easily modifiable to incorporate the infrastructure change to Active Directory if needed.

### IMPORTANT

After configuring the `/etc/samba/smb.conf` file, join the domain *before* starting Samba by typing the following command as **root**:

```
~]# net rpc join -U administrator%password
```

Note that the **-S** option, which specifies the domain server host name, does not need to be stated in the `net rpc join` command. Samba uses the host name specified by the `workgroup` directive in the `/etc/samba/smb.conf` file instead of it being stated explicitly.

#### 21.1.6.3. Domain Controller

A domain controller in Windows NT is functionally similar to a Network Information Service (NIS) server in a Linux environment. Domain controllers and NIS servers both host user and group information databases as well as related services. Domain controllers are mainly used for security, including the authentication of users accessing domain resources. The service that maintains the user and group database integrity is called the *Security Account Manager* (SAM). The SAM database is stored differently between Windows and Linux Samba-based systems, therefore SAM replication cannot be achieved and platforms cannot be mixed in a PDC/BDC environment.

In a Samba environment, there can be only one PDC and zero or more BDCs.

### IMPORTANT

Samba cannot exist in a mixed Samba/Windows domain controller environment (Samba cannot be a BDC of a Windows PDC or vice versa). Alternatively, Samba PDCs and BDCs *can* coexist.

#### Primary Domain Controller (PDC) Using `tldb`

The simplest and most common implementation of a Samba PDC uses the new default `tldb` password database back end. Replacing the aging `smbpasswd` back end, `tldb` has numerous improvements that are explained in more detail in [Section 21.1.8, “Samba Account Information Databases”](#). The `passdb backend` directive controls which back end is to be used for the PDC.

The following `/etc/samba/smb.conf` file shows a sample configuration needed to implement a `tldb` password database back end.

**Example 21.7. An Example Configuration of Primary Domain Controller (PDC) Using tdbsam**

```
[global]
workgroup = DOCS
netbios name = DOCS_SRV
passdb backend = tdbsam
security = user
add user script = /usr/sbin/useradd -m "%u"
delete user script = /usr/sbin/userdel -r "%u"
add group script = /usr/sbin/groupadd "%g"
delete group script = /usr/sbin/groupdel "%g"
add user to group script = /usr/sbin/usermod -G "%g" "%u"
add machine script = /usr/sbin/useradd -s /bin/false -d /dev/null -g
machines "%u"
# The following specifies the default logon script
# Per user logon scripts can be specified in the user
# account using pdbedit logon script = logon.bat
# This sets the default profile path.
# Set per user paths with pdbedit
logon drive = H:
domain logons = yes
os level = 35
preferred master = yes
domain master = yes

[homes]
comment = Home Directories
valid users = %S
read only = no

[netlogon]
comment = Network Logon Service
path = /var/lib/samba/netlogon/scripts
browseable = no
read only = no
# For profiles to work, create a user directory under the
# path shown.
# mkdir -p /var/lib/samba/profiles/john

[Profiles]
comment = Roaming Profile Share
path = /var/lib/samba/profiles
read only = no
browseable = no
guest ok = yes
profile acls = yes
# Other resource shares ... ..
```

To provide a functional PDC system which uses **tdbsam** follow these steps:

1. Adjust the **smb.conf** configuration file as shown in [Example 21.7, “An Example Configuration of Primary Domain Controller \(PDC\) Using tdbsam”](#).

2. Add the **root** user to the Samba password database. You will be prompted to provide a new Samba password for the **root** user:

```
~]# smbpasswd -a root
New SMB password:
```

3. Start the **smb** service:

```
~]# service smb start
```

4. Make sure all profile, user, and netlogon directories are created.
5. Add groups that users can be members of:

```
~]# groupadd -f users
~]# groupadd -f nobody
~]# groupadd -f ntadmins
```

6. Associate the UNIX groups with their respective Windows groups.

```
~]# net groupmap add ntgroup="Domain Users" unixgroup=users
~]# net groupmap add ntgroup="Domain Guests" unixgroup=nobody
~]# net groupmap add ntgroup="Domain Admins" unixgroup=ntadmins
```

7. Grant access rights to a user or a group. For example, to grant the right to add client machines to the domain on a Samba domain controller, to the members to the Domain Admins group, execute the following command:

```
~]# net rpc rights grant 'DOCS\Domain Admins'
SetMachineAccountPrivilege -S PDC -U root
```

Keep in mind that Windows systems prefer to have a primary group which is mapped to a domain group such as Domain Users.

Windows groups and users use the same namespace thus not allowing the existence of a group and a user with the same name like in UNIX.



#### NOTE

If you need more than one domain controller or have more than 250 users, do *not* use the **tdbsam** authentication back end. LDAP is recommended in these cases.

### Primary Domain Controller (PDC) with Active Directory

Although it is possible for Samba to be a member of an Active Directory, it is not possible for Samba to operate as an Active Directory domain controller.

#### 21.1.7. Samba Security Modes

There are only two types of security modes for Samba, *share-level* and *user-level*, which are collectively known as *security levels*. Share-level security is deprecated and Red Hat recommends to use user-level security instead. User-level security can be implemented in one of three different ways. The different ways of implementing a security level are called *security modes*.



### 21.1.7.1. User-Level Security

User-level security is the default and recommended setting for Samba. Even if the ***security = user*** directive is not listed in the ***/etc/samba/smb.conf*** file, it is used by Samba. If the server accepts the client's user name and password, the client can then mount multiple shares without specifying a password for each instance. Samba can also accept session-based user name and password requests. The client maintains multiple authentication contexts by using a unique UID for each logon.

In the ***/etc/samba/smb.conf*** file, the ***security = user*** directive that sets user-level security is:

```
[GLOBAL]
...
security = user
...
```

### Samba Guest Shares

As mentioned above, share-level security mode is deprecated and highly recommended to not use. To configure a Samba guest share without using the ***security = share*** parameter, follow the procedure below:

#### Procedure 21.4. Configuring Samba Guest Shares

1. Create a username map file, in this example ***/etc/samba/smbusers***, and add the following line to it:

```
nobody = guest
```

2. Add the following directives to the main section in the ***/etc/samba/smb.conf*** file. Also, do not use the ***valid users*** directive:

```
[GLOBAL]
...
security = user
map to guest = Bad User
username map = /etc/samba/smbusers
...
```

The ***username map*** directive provides a path to the username map file specified in the previous step.

3. Add the following directive to the share section in the ***/etc/samba/smb.conf*** file. Do not use the ***valid users*** directive.

```
[SHARE]
...
guest ok = yes
...
```

The following sections describe other implementations of user-level security.

### Domain Security Mode (User-Level Security)

In domain security mode, the Samba server has a machine account (domain security trust account) and causes all authentication requests to be passed through to the domain controllers. The Samba server is made into a domain member server by using the following directives in the `/etc/samba/smb.conf` file:

```
[GLOBAL]
...
security = domain
workgroup = MARKETING
...
```

### Active Directory Security Mode (User-Level Security)

If you have an Active Directory environment, it is possible to join the domain as a native Active Directory member. Even if a security policy restricts the use of NT-compatible authentication protocols, the Samba server can join an ADS using Kerberos. Samba in Active Directory member mode can accept Kerberos tickets.

In the `/etc/samba/smb.conf` file, the following directives make Samba an Active Directory member server:

```
[GLOBAL]
...
security = ADS
realm = EXAMPLE.COM
password server = kerberos.example.com
...
```

#### 21.1.7.2. Share-Level Security

With share-level security, the server accepts only a password without an explicit user name from the client. The server expects a password for each share, independent of the user name. There have been recent reports that Microsoft Windows clients have compatibility issues with share-level security servers. This mode is deprecated and Red Hat strongly discourages use of share-level security. Follow steps in [Procedure 21.4, “Configuring Samba Guest Shares”](#) instead of using the `security = share` directive.

#### 21.1.8. Samba Account Information Databases

The following is a list of different back ends you can use with Samba. Other back ends not listed here may also be available.

##### Plain Text

Plain text back ends are nothing more than the `/etc/passwd` type back ends. With a plain text back end, all user names and passwords are sent unencrypted between the client and the Samba server. This method is very insecure and is not recommended for use by any means. It is possible that different Windows clients connecting to the Samba server with plain text passwords cannot support such an authentication method.

##### smbpasswd

The `smbpasswd` back end utilizes a plain ASCII text layout that includes the MS Windows LanMan and NT account, and encrypted password information. The `smbpasswd` back end lacks the storage of the Windows NT/2000/2003 SAM extended controls. The `smbpasswd` back end is not recommended because it does not scale well or hold any Windows information, such as RIDs for NT-based groups. The `tdbsam` back end solves these issues for use in a smaller database (250 users), but is still not an enterprise-class solution.

## ldapsam\_compat

The **ldapsam\_compat** back end allows continued OpenLDAP support for use with upgraded versions of Samba.

## tdbsam

The default **tdbsam** password back end provides a database back end for local servers, servers that do not need built-in database replication, and servers that do not require the scalability or complexity of LDAP. The **tdbsam** back end includes all of the **smbpasswd** database information as well as the previously-excluded SAM information. The inclusion of the extended SAM data allows Samba to implement the same account and system access controls as seen with Windows NT/2000/2003/2008-based systems.

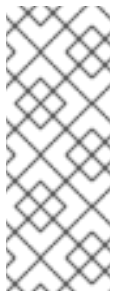
The **tdbsam** back end is recommended for 250 users at most. Larger organizations should require Active Directory or LDAP integration due to scalability and possible network infrastructure concerns.

## ldapsam

The **ldapsam** back end provides an optimal distributed account installation method for Samba. LDAP is optimal because of its ability to replicate its database to any number of servers such as the **Red Hat Directory Server** or an **OpenLDAP Server**. LDAP databases are light-weight and scalable, and as such are preferred by large enterprises. Installation and configuration of directory servers is beyond the scope of this chapter. For more information on the **Red Hat Directory Server**, see the [Red Hat Directory Server 9.0 Deployment Guide](#). For more information on LDAP, see [Section 20.1, “OpenLDAP”](#).

If you are upgrading from a previous version of Samba to 3.0, note that the OpenLDAP schema file (`/usr/share/doc/samba-version/LDAP/samba.schema`) and the Red Hat Directory Server schema file (`/usr/share/doc/samba-version/LDAP/samba-schema-FDS.ldif`) have changed. These files contain the *attribute syntax definitions* and *objectclass definitions* that the **ldapsam** back end needs in order to function properly.

As such, if you are using the **ldapsam** back end for your Samba server, you will need to configure **slapd** to include one of these schema file. See [Section 20.1.3.3, “Extending Schema”](#) for directions on how to do this.



### NOTE

You need to have the `openldap-servers` package installed if you want to use the **ldapsam** back end. To ensure that the package is installed, execute the following command as **roots**:

```
~]# yum install openldap-servers
```

## 21.1.9. 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** host name 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.

### 21.1.9.1. 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 21.1.4, “Configuring a Samba Server”](#)).

### 21.1.9.2. WINS (Windows Internet Name Server)

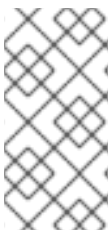
Either a Samba server or a Windows NT server can function as a WINS server. When a WINS server is used with NetBIOS enabled, UDP unicasts can be routed which allows name resolution across networks. Without a WINS server, the UDP broadcast is limited to the local subnet and therefore cannot be routed 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:

#### Example 21.8. An Example Configuration of 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.

### 21.1.10. Samba with CUPS Printing Support

Samba allows client machines to share printers connected to the Samba server. In addition, Samba also allows client machines to send documents built in Linux to Windows printer shares. Although there are other printing systems that function with Red Hat Enterprise Linux, CUPS (Common UNIX Print System) is the recommended printing system due to its close integration with Samba.

#### 21.1.10.1. Simple `smb.conf` Settings

The following example shows a very basic `/etc/samba/smb.conf` configuration for CUPS support:

### Example 21.9. An Example Configuration of Samba with CUPS Support

```
[global]
load printers = yes
printing = cups
printcap name = cups
[printers]
comment = All Printers
path = /var/spool/samba
browseable = no
guest ok = yes
writable = no
printable = yes
printer admin = @ntadmins
[print$]
comment = Printer Drivers Share
path = /var/lib/samba/drivers
write list = ed, john
printer admin = ed, john
```

Other printing configurations are also possible. To add additional security and privacy for printing confidential documents, users can have their own print spooler not located in a public path. If a job fails, other users would not have access to the file.

The `print$` directive contains printer drivers for clients to access if not available locally. The `print$` directive is optional and may not be required depending on the organization.

Setting `browseable` to `yes` enables the printer to be viewed in the Windows Network Neighborhood, provided the Samba server is set up correctly in the domain or workgroup.

## 21.1.11. Samba Distribution Programs

### findsmb

```
findsmb <subnet_broadcast_address>
```

The `findsmb` program is a Perl script which reports information about **SMB**-aware systems on a specific subnet. If no subnet is specified the local subnet is used. Items displayed include **IP** address, NetBIOS name, workgroup or domain name, operating system, and version. The `findsmb` command is used in the following format:

The following example shows the output of executing `findsmb` as any valid user on a system:

```
~]$ findsmb
IP ADDR          NETBIOS NAME  WORKGROUP/OS/VERSION
-----
10.1.59.25       VERVE         [MYGROUP] [Unix] [Samba 3.0.0-15]
10.1.59.26       STATION22    [MYGROUP] [Unix] [Samba 3.0.2-7.FC1]
10.1.56.45       TREK         +[WORKGROUP] [Windows 5.0] [Windows 2000 LAN
Manager]
10.1.57.94       PIXEL        [MYGROUP] [Unix] [Samba 3.0.0-15]
```

```

10.1.57.137  MOBILE001  [WORKGROUP] [Windows 5.0] [Windows 2000 LAN
Manager]
10.1.57.141  JAWS        +[KWIKIMART] [Unix] [Samba 2.2.7a-security-
rollup-fix]
10.1.56.159  FRED        +[MYGROUP] [Unix] [Samba 3.0.0-14.3E]
10.1.59.192  LEGION      *[MYGROUP] [Unix] [Samba 2.2.7-security-rollup-
fix]
10.1.56.205  NANCYN      +[MYGROUP] [Unix] [Samba 2.2.7a-security-
rollup-fix]

```

## 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 the **ftp** utility.

## smbcontrol

```
smbcontrol -i <options>
```

```
smbcontrol <options> <destination> <messagetype> <parameters>
```

The **smbcontrol** program sends control messages to running **smbd**, **nmbd**, or **winbindd** 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** utility, 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.

### 21.1.12. Additional Resources

The following sections give you the means to explore Samba in greater detail.

#### Installed Documentation

- **/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(5)`
  - `samba(7)`
  - `smbd(8)`
  - `nmbd(8)`
  - `winbindd(8)`

#### Related Books

- *The Official Samba-3 HOWTO-Collection* by John H. Terpstra and Jelmer R. Vernooij; Prentice Hall — The official Samba-3 documentation as issued by the Samba development team. This is more of a reference guide than a step-by-step guide.
- *Samba-3 by Example* by John H. Terpstra; Prentice Hall — This is another official release issued by the Samba development team which discusses detailed examples of OpenLDAP, DNS, DHCP, and printing configuration files. This has step-by-step related information that helps in real-world implementations.
- *Using Samba, 2nd Edition* by Jay Ts, Robert Eckstein, and David Collier-Brown; O'Reilly — A good resource for novice to advanced users, which includes comprehensive reference material.

#### 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](http://www.gmane.org), that use the **NNTP** protocol are also available. This an alternative to receiving mailing list emails.

## 21.2. 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.

### 21.2.1. 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 **TLS** 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, see [Chapter 14, 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. See [Section 21.2.2.6.8, “Network Options”](#) for more information about limiting passive ports.

## 21.2.2. 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.

### 21.2.2.1. Starting and Stopping vsftpd

The **vsftpd** RPM installs the **/etc/rc.d/init.d/vsftpd** script, which can be accessed using the **service** command.

To start the server, type the following as **root**:

```
~]# service vsftpd start
```

To stop the server, as type:

```
~]# service vsftpd stop
```

The **restart** option is a shorthand way of stopping and then starting **vsftpd**. This is the most efficient way to make configuration changes take effect after editing the configuration file for **vsftpd**.

To restart the server, as type the following as **root**:

```
~]# service vsftpd restart
```

The **condrestart** (*conditional restart*) option only starts **vsftpd** if it is currently running. This option is useful for scripts, because it does not start the daemon if it is not running. The **try-restart** option is a synonym.

To conditionally restart the server, as root type:

```
~]# service vsftpd condrestart
```

By default, the **vsftpd** service does *not* start automatically at boot time. To configure the **vsftpd** service to start at boot time, use an initscript utility, such as **/sbin/chkconfig**, **/usr/sbin/ntsysv**, or the **Services Configuration Tool** program. See [Chapter 12, Services and Daemons](#) for more information regarding these tools.

### 21.2.2.2. 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, see [Chapter 10, NetworkManager](#). For additional information about network configuration scripts, see [Chapter 11, Network Interfaces](#).

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 [Section 17.2, “BIND”](#).

For **vsftpd** to answer requests on different **IP** addresses, multiple copies of the daemon must be running. In order to make this possible, 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, all configured instances of the **vsftpd** daemon can be started by executing the following command as **root**:

```
~]# service vsftpd start
```

See [Section 21.2.2.1, “Starting and Stopping vsftpd”](#) for a description of other available **service** commands.

Individual instances of the **vsftpd** daemon can be launched from a **root** shell prompt using the following command:

```
~]# vsftpd /etc/vsftpd/configuration-file
```

In the above command, replace *configuration-file* with the unique name of the requested server's configuration file, such as **vsftpd-site-2.conf**.

Other directives to consider altering on a per-server basis are:

- **anon\_root**
- **local\_root**
- **vsftpd\_log\_file**
- **xferlog\_file**

For a detailed list of directives that can be used in the configuration file of the **vsftpd** daemon, see [Section 21.2.2.5, “Files Installed with vsftpd”](#).

### 21.2.2.3. Encrypting vsftpd Connections Using TLS

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 **TLS** protocol to authenticate connections and encrypt all transfers. Note that an **FTP** client that supports **TLS** is needed to communicate with **vsftpd** with **TLS** enabled.



#### NOTE

**SSL** (Secure Sockets Layer) is the name of an older implementation of the security protocol. The new versions are called **TLS** (Transport Layer Security). Only the newer versions (**TLS**) should be used as **SSL** suffers from serious security vulnerabilities. The documentation included with the **vsftpd** server, as well as the configuration directives used in the **vsftpd.conf** file, use the **SSL** name when referring to security-related matters, but **TLS** is supported and used by default when the **ssl\_enable** directive is set to **YES**.

Set the **ssl\_enable** configuration directive in the **vsftpd.conf** file to **YES** to turn on **TLS** support. The default settings of other **TLS**-related directives that become automatically active when the **ssl\_enable** option is enabled provide for a reasonably well-configured **TLS** set up. This includes, among other

things, the requirement to only use the **TLS** v1 protocol for all connections (the use of the insecure **SSL** protocol versions is disabled by default) or forcing all non-anonymous logins to use **TLS** for sending passwords and data transfers.

### Example 21.10. Configuring vsftpd to Use TLS

In this example, the configuration directives explicitly disable the older **SSL** versions of the security protocol in the **vsftpd.conf** file:

```
ssl_enable=YES
ssl_tlsv1=YES
ssl_sslv2=NO
ssl_sslv3=NO
```

Restart the **vsftpd** service after you modify its configuration:

```
~]# service vsftpd restart
```

See the `vsftpd.conf(5)` manual page for other **TLS**-related configuration directives for fine-tuning the use of **TLS** by **vsftpd**. Also, see [Section 21.2.2.6, “vsftpd Configuration Options”](#) for a description of other commonly used **vsftpd.conf** configuration directives.

#### 21.2.2.4. 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 want 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 6, 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 6, 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 6 Security-Enhanced Linux](#) for more detailed information about SELinux in general.

#### 21.2.2.5. Files Installed with vsftpd



The `vsftpd` RPM installs the daemon (**`vsftpd`**), its configuration and related files, as well as **FTP** directories onto the system. The following lists the files and directories related to **`vsftpd`** configuration:

- **`/etc/pam.d/vsftpd`** — The *Pluggable Authentication Modules* (PAM) configuration file for **`vsftpd`**. This file specifies the requirements a user must meet to log in to the **FTP** server. For more information on PAM, see the *Using Pluggable Authentication Modules (PAM)* chapter of the [Red Hat Enterprise Linux 6 Single Sign-On and Smart Cards](#) guide.
- **`/etc/vsftpd/vsftpd.conf`** — The configuration file for **`vsftpd`**. See [Section 21.2.2.6, “vsftpd Configuration Options”](#) for a list of important options contained within this file.
- **`/etc/vsftpd/ftpusers`** — A list of users not allowed to log in to **`vsftpd`**. By default, this list includes the **`root`**, **`bin`**, and **`daemon`** users, among others.
- **`/etc/vsftpd/user_list`** — This file can be configured to either deny or allow access to the users listed, depending on whether the **`userlist_deny`** directive is set to **YES** (default) or **NO** in **`/etc/vsftpd/vsftpd.conf`**. If **`/etc/vsftpd/user_list`** is used to grant access to users, the user names listed must *not* appear in **`/etc/vsftpd/ftpusers`**.
- **`/var/ftp/`** — The directory containing files served by **`vsftpd`**. It also contains the **`/var/ftp/pub/`** directory for anonymous users. Both directories are world-readable, but writable only by the **`root`** user.

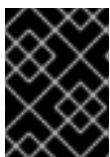
### 21.2.2.6. vsftpd Configuration Options

Although **`vsftpd`** may not offer the level of customization other widely available **FTP** servers have, it offers enough options to satisfy most administrators' needs. The fact that it is not overly feature-laden limits configuration and programmatic errors.

All configuration of **`vsftpd`** is handled by its configuration file, **`/etc/vsftpd/vsftpd.conf`**. Each directive is on its own line within the file and follows the following format:

```
directive=value
```

For each directive, replace *directive* with a valid directive and *value* with a valid value.



#### IMPORTANT

There must not be any spaces between the *directive*, equal symbol, and the *value* in a directive.

Comment lines must be preceded by a hash symbol (**`#`**) and are ignored by the daemon.

For a complete list of all directives available, see the man page for **`vsftpd.conf`**. For an overview of ways to secure **`vsftpd`**, see the [Red Hat Enterprise Linux 6 Security Guide](#).

The following is a list of some of the more important directives within **`/etc/vsftpd/vsftpd.conf`**. All directives not explicitly found or commented out within the **`vsftpd`**'s configuration file are set to their default value.

#### 21.2.2.6.1. Daemon Options

The following is a list of directives that control the overall behavior of the **`vsftpd`** daemon.



- **listen** — When enabled, **vsftpd** runs in standalone mode, which means that the daemon is started independently, not by the **xinetd** super-server. Red Hat Enterprise Linux 6 sets this value to **YES**. Note that the SELinux **ftpd\_is\_daemon** Boolean option needs to be set for **vsftpd** to be allowed to run in standalone mode. See [Section 21.2.2.4, “SELinux Policy for vsftpd”](#) and to **ftpd\_selinux(8)** for more information on **vsftpd**'s interaction with the default SELinux policy. This directive cannot be used in conjunction with the **listen\_ipv6** directive.

The default value is **NO**. On Red Hat Enterprise Linux 6, this option is set to **YES** in the configuration file.

- **listen\_ipv6** — When enabled, **vsftpd** runs in standalone mode, which means that the daemon is started independently, not by the **xinetd** super-server. With this directive, it only listens on **IPv6** sockets. Note that the SELinux **ftpd\_is\_daemon** Boolean option needs to be set for **vsftpd** to be allowed to run in standalone mode. See [Section 21.2.2.4, “SELinux Policy for vsftpd”](#) and to **ftpd\_selinux(8)** for more information on **vsftpd**'s interaction with the default SELinux policy. This directive cannot be used in conjunction with the **listen** directive.

The default value is **NO**.

- **session\_support** — When enabled, **vsftpd** attempts to maintain login sessions for each user through *Pluggable Authentication Modules* (PAM). For more information, see the *Using Pluggable Authentication Modules (PAM)* chapter of the [Red Hat Enterprise Linux 6 Single Sign-On and Smart Cards](#) and the PAM man pages. If session logging is not necessary, disabling this option allows **vsftpd** to run with less processes and lower privileges.

The default value is **YES**.

#### 21.2.2.6.2. Log In Options and Access Controls

The following is a list of directives that control the login behavior and access-control mechanisms.

- **anonymous\_enable** — When enabled, anonymous users are allowed to log in. The user names **anonymous** and **ftp** are accepted.

The default value is **YES**.

See [Section 21.2.2.6.3, “Anonymous User Options”](#) for a list of directives affecting anonymous users.

- **banned\_email\_file** — If the **deny\_email\_enable** directive is set to **YES**, this directive specifies the file containing a list of anonymous email passwords that are not permitted access to the server.

The default value is **/etc/vsftpd/banned\_emails**.

- **banner\_file** — Specifies the file containing text displayed when a connection is established to the server. This option overrides any text specified in the **ftpd\_banner** directive.

There is no default value for this directive.

- **cmds\_allowed** — Specifies a comma-delimited list of **FTP** commands allowed by the server. All other commands are rejected.

There is no default value for this directive.

- **deny\_email\_enable** — When enabled, any anonymous user utilizing email passwords specified in `/etc/vsftpd/banned_emails` are denied access to the server. The name of the file referenced by this directive can be specified using the **banned\_email\_file** directive.

The default value is **NO**.

- **ftpd\_banner** — When enabled, the string specified within this directive is displayed when a connection is established to the server. This option can be overridden by the **banner\_file** directive.

By default, **vsftpd** displays its standard banner.

- **local\_enable** — When enabled, local users are allowed to log in to the system. Note that the SELinux **ftp\_home\_dir** Boolean option needs to be set for this directive to work as expected. See [Section 21.2.2.4, “SELinux Policy for vsftpd”](#) and to **ftpd\_selinux(8)** for more information on **vsftpd**'s interaction with the default SELinux policy.

The default value is **NO**. On Red Hat Enterprise Linux 6, this option is set to **YES** in the configuration file.

See [Section 21.2.2.6.4, “Local-User Options”](#) for a list of directives affecting local users.

- **pam\_service\_name** — Specifies the PAM service name for **vsftpd**.

The default value is **ftp**. On Red Hat Enterprise Linux 6, this option is set to **vsftpd** in the configuration file.

- **tcp\_wrappers** — When enabled, TCP wrappers are used to grant access to the server. If the FTP server is configured on multiple IP addresses, the **VSFTPD\_LOAD\_CONF** environment variable can be used to load different configuration files based on the IP address being requested by the client.

The default value is **NO**. On Red Hat Enterprise Linux 6, this option is set to **YES** in the configuration file.

- **userlist\_deny** — When used in conjunction with the **userlist\_enable** directive and set to **NO**, all local users are denied access unless their user name is listed in the file specified by the **userlist\_file** directive. Because access is denied before the client is asked for a password, setting this directive to **NO** prevents local users from submitting unencrypted passwords over the network.

The default value is **YES**.

- **userlist\_enable** — When enabled, users listed in the file specified by the **userlist\_file** directive are denied access. Because access is denied before the client is asked for a password, users are prevented from submitting unencrypted passwords over the network.

The default value is **NO**. On Red Hat Enterprise Linux 6, this option is set to **YES** in the configuration file.

- **userlist\_file** — Specifies the file referenced by **vsftpd** when the **userlist\_enable** directive is enabled.

The default value is `/etc/vsftpd/user_list`, which is created during installation.

### 21.2.2.6.3. Anonymous User Options

The following lists directives that control anonymous user access to the server. To use these options, the **anonymous\_enable** directive must be set to **YES**.

- **anon\_mkdir\_write\_enable** — When enabled in conjunction with the **write\_enable** directive, anonymous users are allowed to create new directories within a parent directory that has write permissions.

The default value is **NO**.

- **anon\_root** — Specifies the directory **vsftpd** changes to after an anonymous user logs in.

There is no default value for this directive.

- **anon\_upload\_enable** — When enabled in conjunction with the **write\_enable** directive, anonymous users are allowed to upload files within a parent directory that has write permissions.

The default value is **NO**.

- **anon\_world\_readable\_only** — When enabled, anonymous users are only allowed to download world-readable files.

The default value is **YES**.

- **ftp\_username** — Specifies the local user account (listed in **/etc/passwd**) used for the anonymous **FTP** user. The home directory specified in **/etc/passwd** for the user is the root directory of the anonymous **FTP** user.

The default value is **ftp**.

- **no\_anon\_password** — When enabled, the anonymous user is not asked for a password.

The default value is **NO**.

- **secure\_email\_list\_enable** — When enabled, only a specified list of email passwords for anonymous logins is accepted. This is a convenient way of offering limited security to public content without the need for virtual users.

Anonymous logins are prevented unless the password provided is listed in **/etc/vsftpd/email\_passwords**. The file format is one password per line, with no trailing white spaces.

The default value is **NO**.

### 21.2.2.6.4. Local-User Options

The following lists directives that characterize the way local users access the server. To use these options, the **local\_enable** directive must be set to **YES**. Note that the SELinux **ftp\_home\_dir** Boolean option needs to be set for users to be able to access their home directories. See [Section 21.2.2.4, “SELinux Policy for vsftpd”](#) and to **ftpd\_selinux(8)** for more information on **vsftpd**'s interaction with the default SELinux policy.

- **chmod\_enable** — When enabled, the **FTP** command **SITE CHMOD** is allowed for local users. This command allows the users to change the permissions on files.

The default value is **YES**.

- **chroot\_list\_enable** — When enabled, the local users listed in the file specified in the **chroot\_list\_file** directive are placed in a **chroot** jail upon log in.

If enabled in conjunction with the **chroot\_local\_user** directive, the local users listed in the file specified in the **chroot\_list\_file** directive are *not* placed in a **chroot** jail upon log in.

The default value is **NO**.

- **chroot\_list\_file** — Specifies the file containing a list of local users referenced when the **chroot\_list\_enable** directive is set to **YES**.

The default value is `/etc/vsftpd/chroot_list`.

- **chroot\_local\_user** — When enabled, local users are change-rooted to their home directories after logging in.

The default value is **NO**.



### WARNING

Enabling **chroot\_local\_user** opens up a number of security issues, especially for users with upload privileges. For this reason, it is *not* recommended.

- **guest\_enable** — When enabled, all non-anonymous users are logged in as the user **guest**, which is the local user specified in the **guest\_username** directive.

The default value is **NO**.

- **guest\_username** — Specifies the user name the **guest** user is mapped to.

The default value is **ftp**.

- **local\_root** — Specifies the directory **vsftpd** changes to after a local user logs in.

There is no default value for this directive.

- **local\_umask** — Specifies the umask value for file creation. Note that the default value is in octal form (a numerical system with a base of eight), which includes a “0” prefix. Otherwise, the value is treated as a base-10 integer.

The default value is **077**. On Red Hat Enterprise Linux 6, this option is set to **022** in the configuration file.

- **passwd\_chroot\_enable** — When enabled in conjunction with the **chroot\_local\_user** directive, **vsftpd** change-roots local users based on the occurrence of `/./` in the home-directory field within `/etc/passwd`.

The default value is **NO**.

- **user\_config\_dir** — Specifies the path to a directory containing configuration files bearing the names of local system users that contain specific settings for those users. Any directive in a user's configuration file overrides those found in `/etc/vsftpd/vsftpd.conf`.

There is no default value for this directive.

#### 21.2.2.6.5. Directory Options

The following lists directives that affect directories.

- **dirlist\_enable** — When enabled, users are allowed to view directory lists.

The default value is **YES**.

- **dirmessage\_enable** — When enabled, a message is displayed whenever a user enters a directory with a message file. This message resides within the current directory. The name of this file is specified in the **message\_file** directive and is **.message** by default.

The default value is **NO**. On Red Hat Enterprise Linux 6, this option is set to **YES** in the configuration file.

- **force\_dot\_files** — When enabled, files beginning with a dot ( `.` ) are listed in directory listings, with the exception of the `.` and `..` files.

The default value is **NO**.

- **hide\_ids** — When enabled, all directory listings show **ftp** as the user and group for each file.

The default value is **NO**.

- **message\_file** — Specifies the name of the message file when using the **dirmessage\_enable** directive.

The default value is **.message**.

- **text\_userdb\_names** — When enabled, text user names and group names are used in place of UID and GID entries. Enabling this option may negatively affect the performance of the server.

The default value is **NO**.

- **use\_localtime** — When enabled, directory listings reveal the local time for the computer instead of GMT.

The default value is **NO**.

#### 21.2.2.6.6. File Transfer Options

The following lists directives that affect directories.

- **download\_enable** — When enabled, file downloads are permitted.

The default value is **YES**.

- **chown\_uploads** — When enabled, all files uploaded by anonymous users are owned by the

user specified in the **chown\_username** directive.

The default value is **NO**.

- **chown\_username** — Specifies the ownership of anonymously uploaded files if the **chown\_uploads** directive is enabled.

The default value is **root**.

- **write\_enable** — When enabled, **FTP** commands which can change the file system are allowed, such as **DELE**, **RNFR**, and **STOR**.

The default value is **NO**. On Red Hat Enterprise Linux 6, this option is set to **YES** in the configuration file.

### 21.2.2.6.7. Logging Options

The following lists directives that affect **vsftpd**'s logging behavior.

- **dual\_log\_enable** — When enabled in conjunction with **xferlog\_enable**, **vsftpd** writes two files simultaneously: a **wu-ftp**-compatible log to the file specified in the **xferlog\_file** directive (**/var/log/xferlog** by default) and a standard **vsftpd** log file specified in the **vsftpd\_log\_file** directive (**/var/log/vsftpd.log** by default).

The default value is **NO**.

- **log\_ftp\_protocol** — When enabled in conjunction with **xferlog\_enable** and with **xferlog\_std\_format** set to **NO**, all **FTP** commands and responses are logged. This directive is useful for debugging.

The default value is **NO**.

- **syslog\_enable** — When enabled in conjunction with **xferlog\_enable**, all logging normally written to the standard **vsftpd** log file specified in the **vsftpd\_log\_file** directive (**/var/log/vsftpd.log** by default) is sent to the system logger instead under the **FTPD** facility.

The default value is **NO**.

- **vsftpd\_log\_file** — Specifies the **vsftpd** log file. For this file to be used, **xferlog\_enable** must be enabled and **xferlog\_std\_format** must either be set to **NO** or, if **xferlog\_std\_format** is set to **YES**, **dual\_log\_enable** must be enabled. It is important to note that if **syslog\_enable** is set to **YES**, the system log is used instead of the file specified in this directive.

The default value is **/var/log/vsftpd.log**.

- **xferlog\_enable** — When enabled, **vsftpd** logs connections (**vsftpd** format only) and file-transfer information to the log file specified in the **vsftpd\_log\_file** directive (**/var/log/vsftpd.log** by default). If **xferlog\_std\_format** is set to **YES**, file-transfer information is logged, but connections are not, and the log file specified in **xferlog\_file** (**/var/log/xferlog** by default) is used instead. It is important to note that both log files and log formats are used if **dual\_log\_enable** is set to **YES**.

The default value is **NO**. On Red Hat Enterprise Linux 6, this option is set to **YES** in the configuration file.

- **xferlog\_file** — Specifies the **wu-ftpd**-compatible log file. For this file to be used, **xferlog\_enable** must be enabled and **xferlog\_std\_format** must be set to **YES**. It is also used if **dual\_log\_enable** is set to **YES**.

The default value is `/var/log/xferlog`.

- **xferlog\_std\_format** — When enabled in conjunction with **xferlog\_enable**, only a **wu-ftpd**-compatible file-transfer log is written to the file specified in the **xferlog\_file** directive (`/var/log/xferlog` by default). It is important to note that this file only logs file transfers and does not log connections to the server.

The default value is **NO**. On Red Hat Enterprise Linux 6, this option is set to **YES** in the configuration file.



### IMPORTANT

To maintain compatibility with log files written by the older **wu-ftpd FTP** server, the **xferlog\_std\_format** directive is set to **YES** under Red Hat Enterprise Linux 6. However, this setting means that connections to the server are not logged. To both log connections in **vsftpd** format and maintain a **wu-ftpd**-compatible file-transfer log, set **dual\_log\_enable** to **YES**. If maintaining a **wu-ftpd**-compatible file-transfer log is not important, either set **xferlog\_std\_format** to **NO**, comment the line with a hash symbol (“#”), or delete the line entirely.

#### 21.2.2.6.8. Network Options

The following lists directives that define how **vsftpd** interacts with the network.

- **accept\_timeout** — Specifies the amount of time for a client using passive mode to establish a connection.

The default value is **60**.

- **anon\_max\_rate** — Specifies the maximum data transfer rate for anonymous users in bytes per second.

The default value is **0**, which does not limit the transfer rate.

- **connect\_from\_port\_20** — When enabled, **vsftpd** runs with enough privileges to open port **20** on the server during active-mode data transfers. Disabling this option allows **vsftpd** to run with less privileges but may be incompatible with some **FTP** clients.

The default value is **NO**. On Red Hat Enterprise Linux 6, this option is set to **YES** in the configuration file.

- **connect\_timeout** — Specifies the maximum amount of time a client using active mode has to respond to a data connection, in seconds.

The default value is **60**.

- **data\_connection\_timeout** — Specifies maximum amount of time data transfers are allowed to stall, in seconds. Once triggered, the connection to the remote client is closed.

The default value is **300**.

- **ftp\_data\_port** — Specifies the port used for active data connections when **connect\_from\_port\_20** is set to **YES**.

The default value is **20**.

- **idle\_session\_timeout** — Specifies the maximum amount of time between commands from a remote client. Once triggered, the connection to the remote client is closed.

The default value is **300**.

- **listen\_address** — Specifies the **IP** address on which **vsftpd** listens for network connections.

There is no default value for this directive.



#### NOTE

If running multiple copies of **vsftpd** serving different **IP** addresses, the configuration file for each copy of the **vsftpd** daemon must have a different value for this directive. See [Section 21.2.2.2, “Starting Multiple Copies of vsftpd”](#) for more information about multihomed **FTP** servers.

- **listen\_address6** — Specifies the **IPv6** address on which **vsftpd** listens for network connections when **listen\_ipv6** is set to **YES**.

There is no default value for this directive.



#### NOTE

If running multiple copies of **vsftpd** serving different **IP** addresses, the configuration file for each copy of the **vsftpd** daemon must have a different value for this directive. See [Section 21.2.2.2, “Starting Multiple Copies of vsftpd”](#) for more information about multihomed **FTP** servers.

- **listen\_port** — Specifies the port on which **vsftpd** listens for network connections.

The default value is **21**.

- **local\_max\_rate** — Specifies the maximum rate at which data is transferred for local users logged in to the server in bytes per second.

The default value is **0**, which does not limit the transfer rate.

- **max\_clients** — Specifies the maximum number of simultaneous clients allowed to connect to the server when it is running in standalone mode. Any additional client connections would result in an error message.

The default value is **0**, which does not limit connections.



- **max\_per\_ip** — Specifies the maximum number of clients allowed to connect from the same source **IP** address.

The default value is **50**. The value **0** switches off the limit.

- **pasv\_address** — Specifies the **IP** address for the public-facing **IP** address of the server for servers behind *Network Address Translation* (NAT) firewalls. This enables **vsftpd** to hand out the correct return address for passive-mode connections.

There is no default value for this directive.

- **pasv\_enable** — When enabled, passive-mode connections are allowed.

The default value is **YES**.

- **pasv\_max\_port** — Specifies the highest possible port sent to **FTP** clients for passive-mode connections. This setting is used to limit the port range so that firewall rules are easier to create.

The default value is **0**, which does not limit the highest passive-port range. The value must not exceed **65535**.

- **pasv\_min\_port** — Specifies the lowest possible port sent to **FTP** clients for passive-mode connections. This setting is used to limit the port range so that firewall rules are easier to create.

The default value is **0**, which does not limit the lowest passive-port range. The value must not be lower than **1024**.

- **pasv\_promiscuous** — When enabled, data connections are not checked to make sure they are originating from the same **IP** address. This setting is only useful for certain types of tunneling.



#### WARNING

Do not enable this option unless absolutely necessary as it disables an important security feature, which verifies that passive-mode connections originate from the same **IP** address as the control connection that initiates the data transfer.

The default value is **NO**.

- **port\_enable** — When enabled, active-mode connects are allowed.

The default value is **YES**.

#### 21.2.2.6.9. Security Options

The following lists directives that can be used to improve **vsftpd** security.

- **isolate\_network** — If enabled, **vsftpd** uses the **CLONE\_NEWNET** container flag to isolate the unprivileged protocol handler processes, so that they cannot arbitrarily call **connect()** and

instead have to ask the privileged process for sockets (the **port\_promiscuous** option must be disabled).

The default value is **YES**.

- **isolate** — If enabled, **vsftpd** uses the **CLONE\_NEWPID** and **CLONE\_NEWIPC** container flags to isolate processes to their IPC and PID namespaces to prevent them from interacting with each other.

The default value is **YES**.

- **ssl\_enable** — Enables **vsftpd**'s support for **SSL** (including **TLS**). **SSL** is used both for authentication and subsequent data transfers. Note that all other **SSL**-related options are only applicable if **ssl\_enable** is set to **YES**.

The default value is **NO**.

- **allow\_anon\_ssl** — Specifies whether anonymous users should be allowed to use secured **SSL** connections.

The default value is **NO**.

- **require\_cert** — If enabled, all **SSL** client connections are required to present a client certificate.

The default value is **NO**.

### 21.2.3. Additional Resources

For more information about **vsftpd** configuration, see the following resources.

#### 21.2.3.1. Installed Documentation

- The `/usr/share/doc/vsftpd-version-number/` directory — The **TUNING** file contains basic performance-tuning tips and the **SECURITY/** directory contains information about the security model employed by **vsftpd**.
- **vsftpd**-related man pages — There are a number of man pages for the daemon and the configuration files. The following lists some of the more important man 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

- `man ftpd_selinux` — 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.

### 21.2.3.2. 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 6 Security-Enhanced Linux](#) — The *Security-Enhanced Linux* for Red Hat Enterprise Linux 6 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 6 Security Guide](#) — The *Security Guide* for Red Hat Enterprise Linux 6 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.

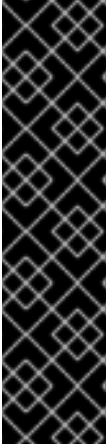
#### Relevant RFC Documents

- [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.

## 21.3. 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.

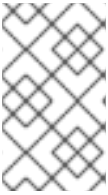
The tool is based on the Common Unix Printing System (CUPS). If you upgraded the system from a previous Red Hat Enterprise Linux version that used CUPS, the upgrade process preserved the configured printers.



## IMPORTANT

The `cupsd.conf` man page documents configuration of a CUPS server. It includes directives for enabling **SSL** support. However, CUPS does not allow control of the protocol versions used. Due to the vulnerability described in [Resolution for POODLE SSLv3.0 vulnerability \(CVE-2014-3566\) for components that do not allow SSLv3 to be disabled via configuration settings](#), Red Hat recommends that you do not rely on this for security. It is recommended that you use **stunnel** to provide a secure tunnel and disable **SSLv3**. For more information on using **stunnel**, see the [Red Hat Enterprise Linux 6 Security Guide](#).

For ad-hoc secure connections to a remote system's **Print Settings** tool, use X11 forwarding over **SSH** as described in [Section 14.5.1, "X11 Forwarding"](#).



## NOTE

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 see the links on the **Home** tab of the web site.

### 21.3.1. 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 21.3, "Printer Configuration window"](#) appears.

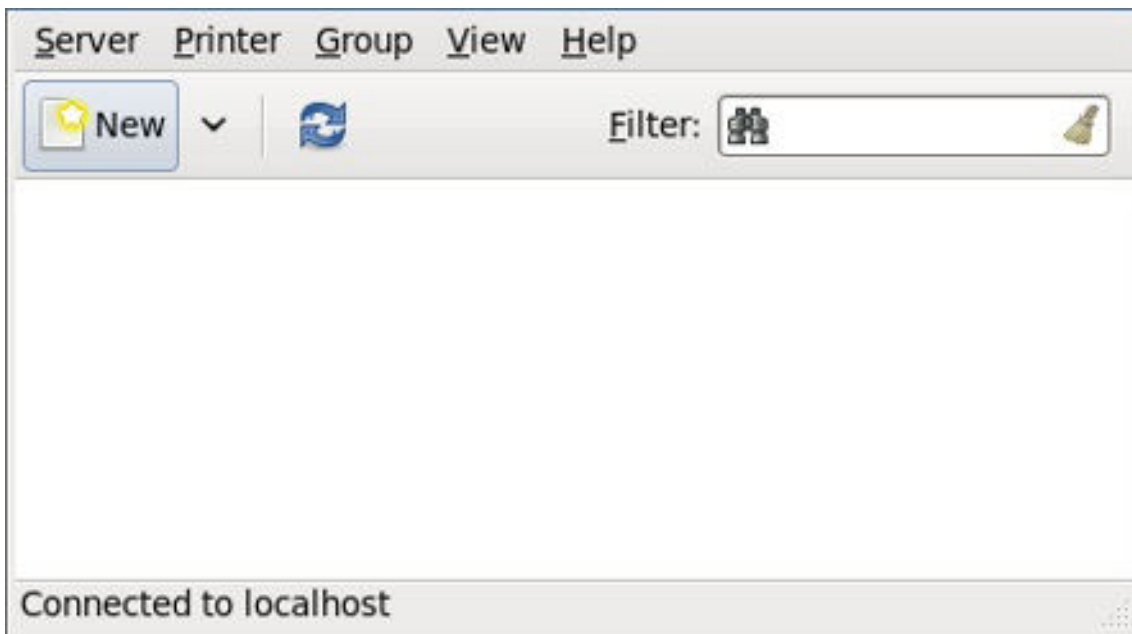


Figure 21.3. Printer Configuration window

### 21.3.2. 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 (see [Section 21.3.1, “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.

### 21.3.3. 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 (see [Section 21.3.2, “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)

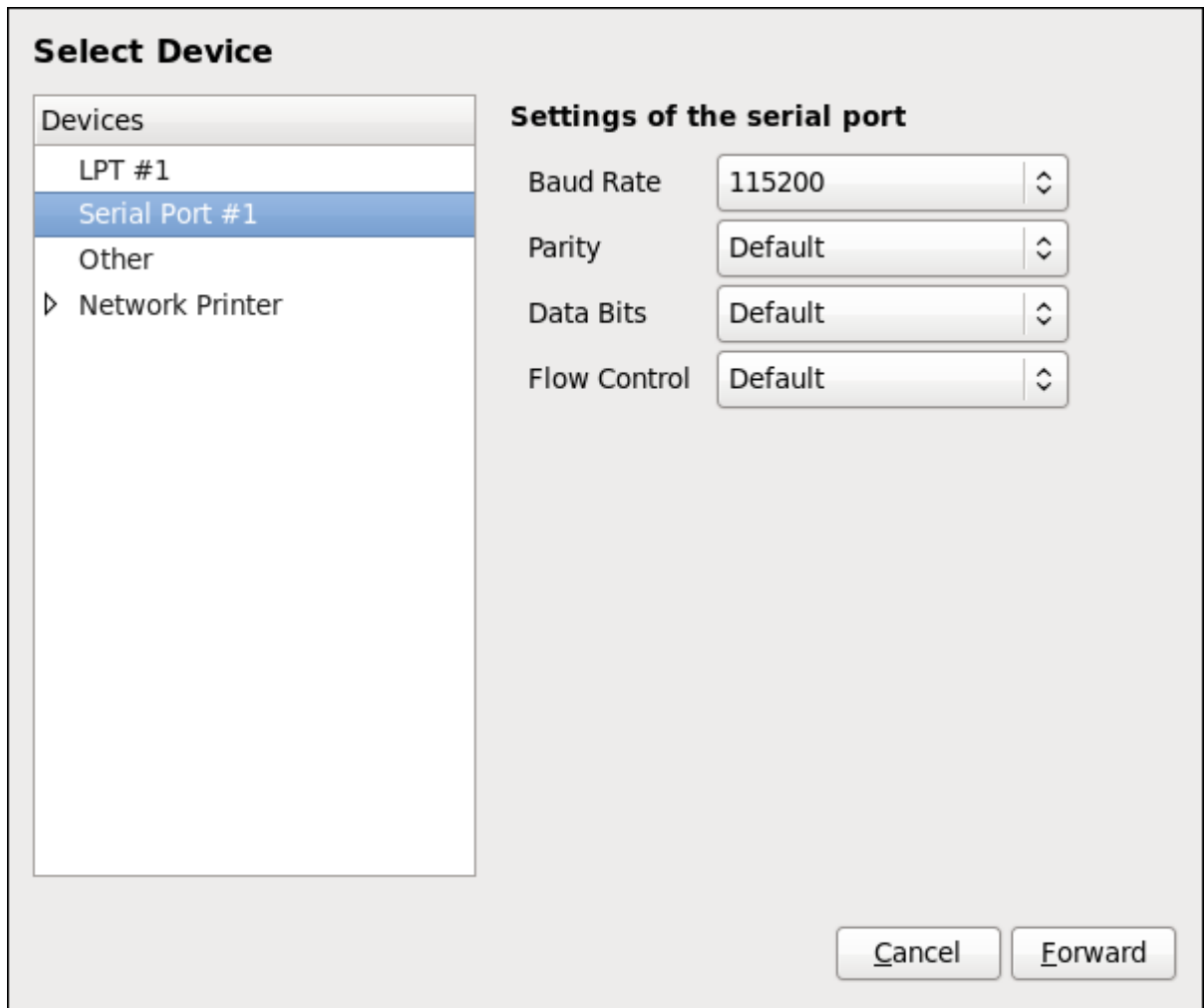
**for Serial Port**

Baud Rate

Parity

Data Bits

Flow Control



**Figure 21.4. Adding a local printer**

4. Click **Forward**.
5. Select the printer model. See [Section 21.3.8, “Selecting the Printer Model and Finishing”](#) for details.

### 21.3.4. Adding an AppSocket/HP JetDirect printer

Follow this procedure to add an AppSocket/HP JetDirect printer:

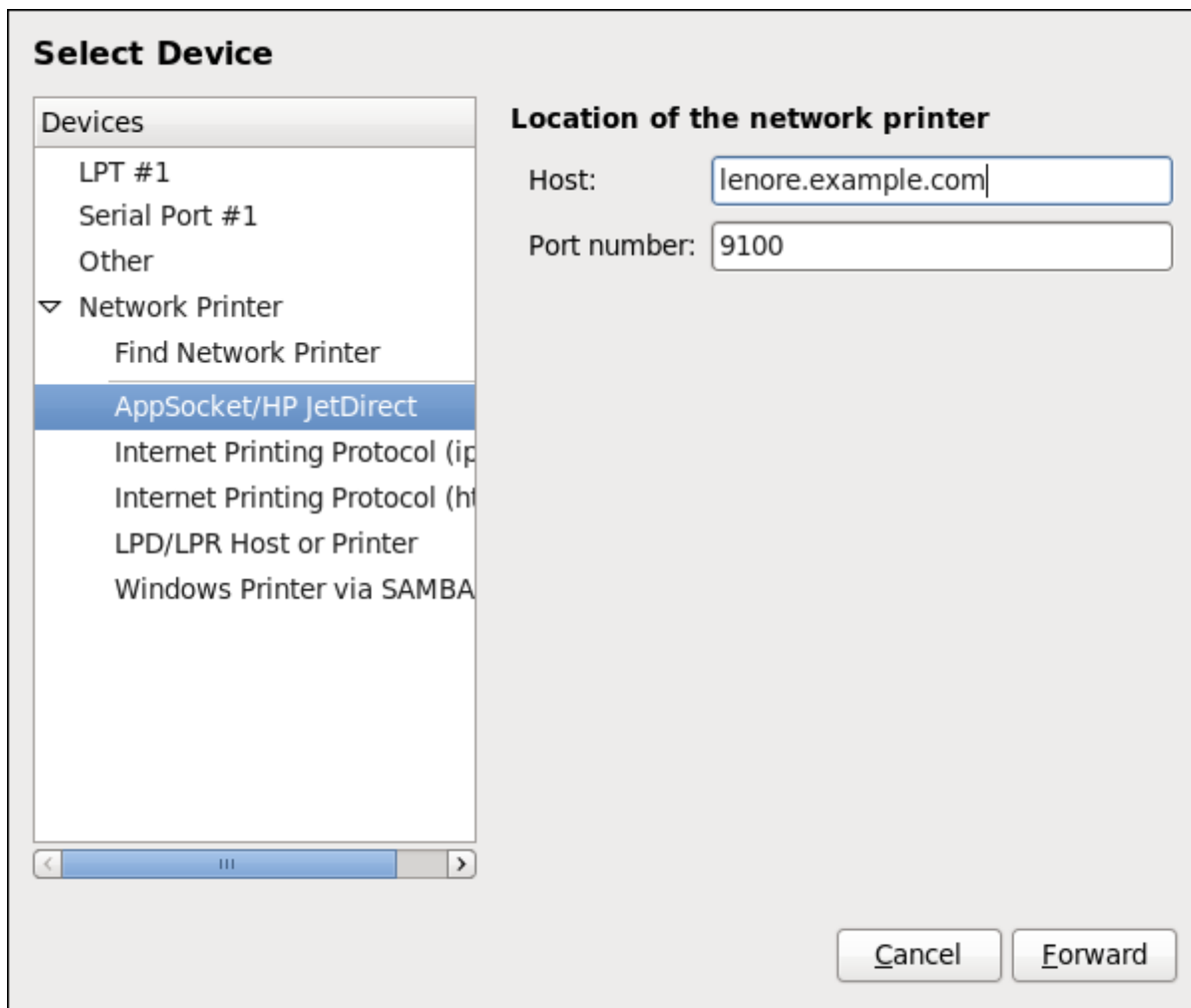
1. Open the **New Printer** dialog (see [Section 21.3.1, “Starting the Printer Configuration Tool”](#)).
2. In the list on the left, select **Network Printer** → **AppSocket/HP JetDirect**.
3. On the right, enter the connection settings:

**Hostname**

Printer host name or IP address.

**Port Number**

Printer port listening for print jobs (**9100** by default).



**Figure 21.5. Adding a JetDirect printer**

4. Click **Forward**.
5. Select the printer model. See [Section 21.3.8, “Selecting the Printer Model and Finishing”](#) for details.

### 21.3.5. 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 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:

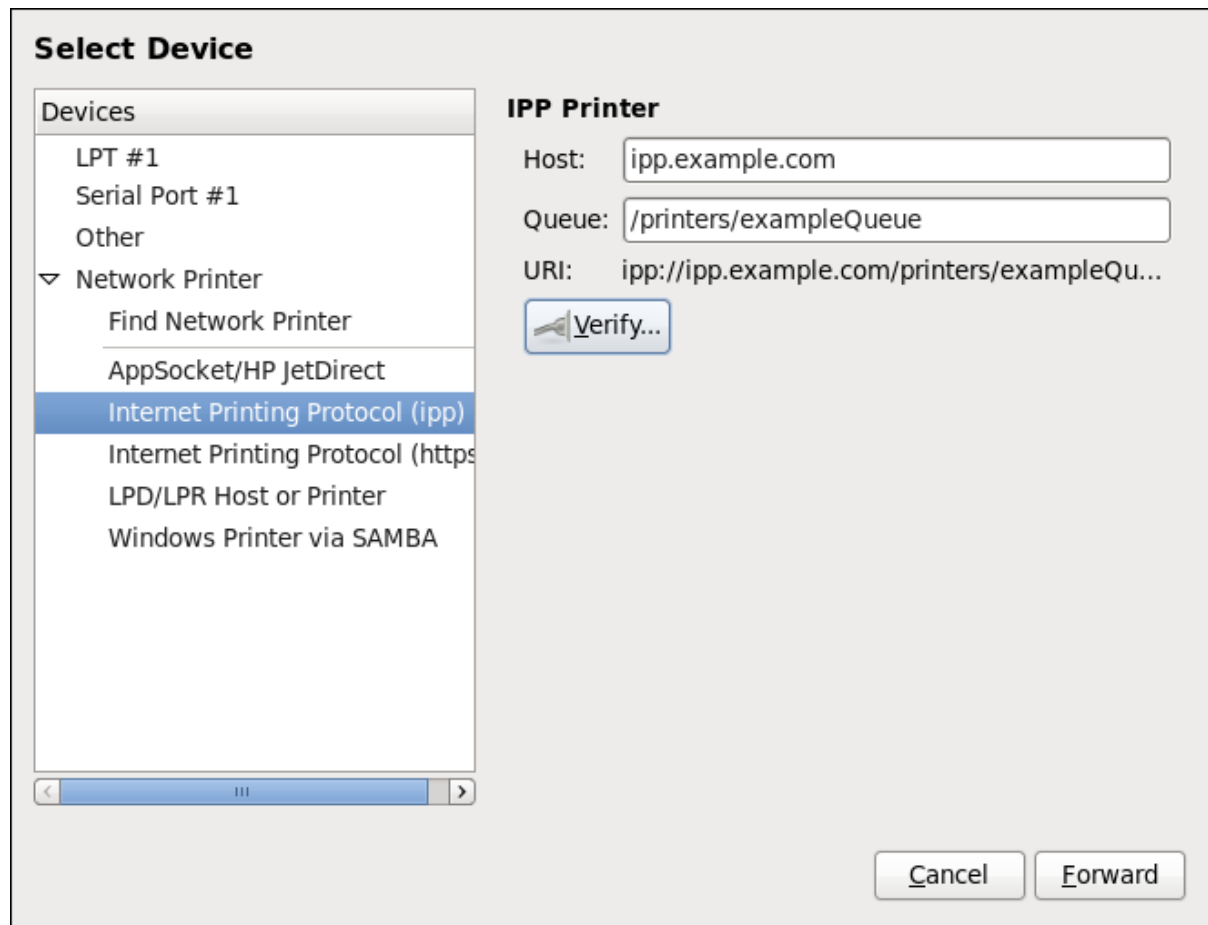
1. Open the **New Printer** dialog (see [Section 21.3.2, “Starting Printer Setup”](#)).
2. In the list of devices on the left, select **Network Printer** and **Internet Printing Protocol (ipp)** or **Internet Printing Protocol (https)**.
3. On the right, enter the connection settings:

**Host**

The host name 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 21.6. Adding an IPP printer**

4. Click **Forward** to continue.
5. Select the printer model. See [Section 21.3.8, “Selecting the Printer Model and Finishing”](#) for details.

### 21.3.6. Adding an LPD/LPR Host or Printer

Follow this procedure to add an LPD/LPR host or printer:

1. Open the **New Printer** dialog (see [Section 21.3.2, “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

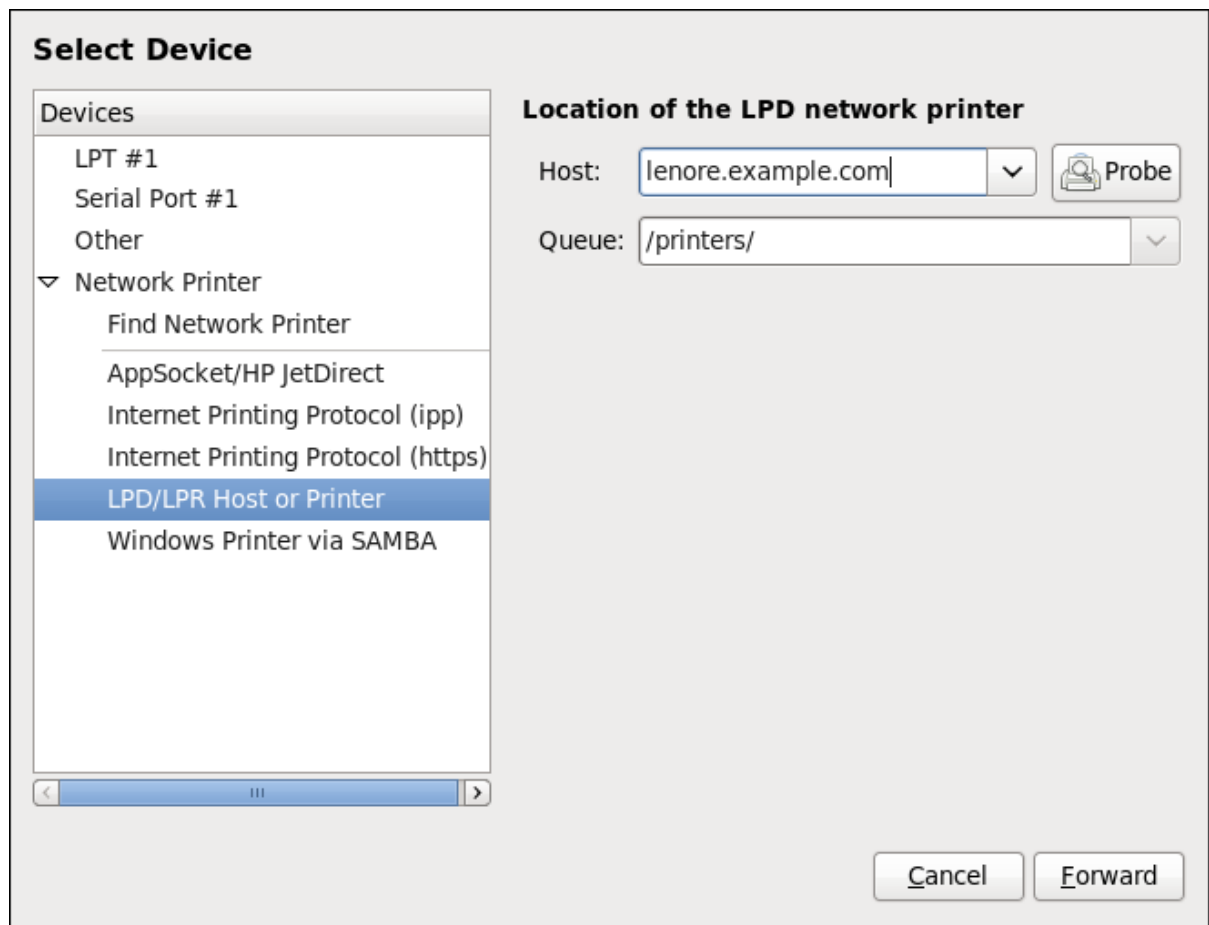
The host name of the LPD/LPR printer or host.

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 21.7. Adding an LPD/LPR printer**

4. Click **Forward** to continue.
5. Select the printer model. See [Section 21.3.8, “Selecting the Printer Model and Finishing”](#) for details.

### 21.3.7. Adding a Samba (SMB) printer

Follow this procedure to add a Samba printer:



#### NOTE

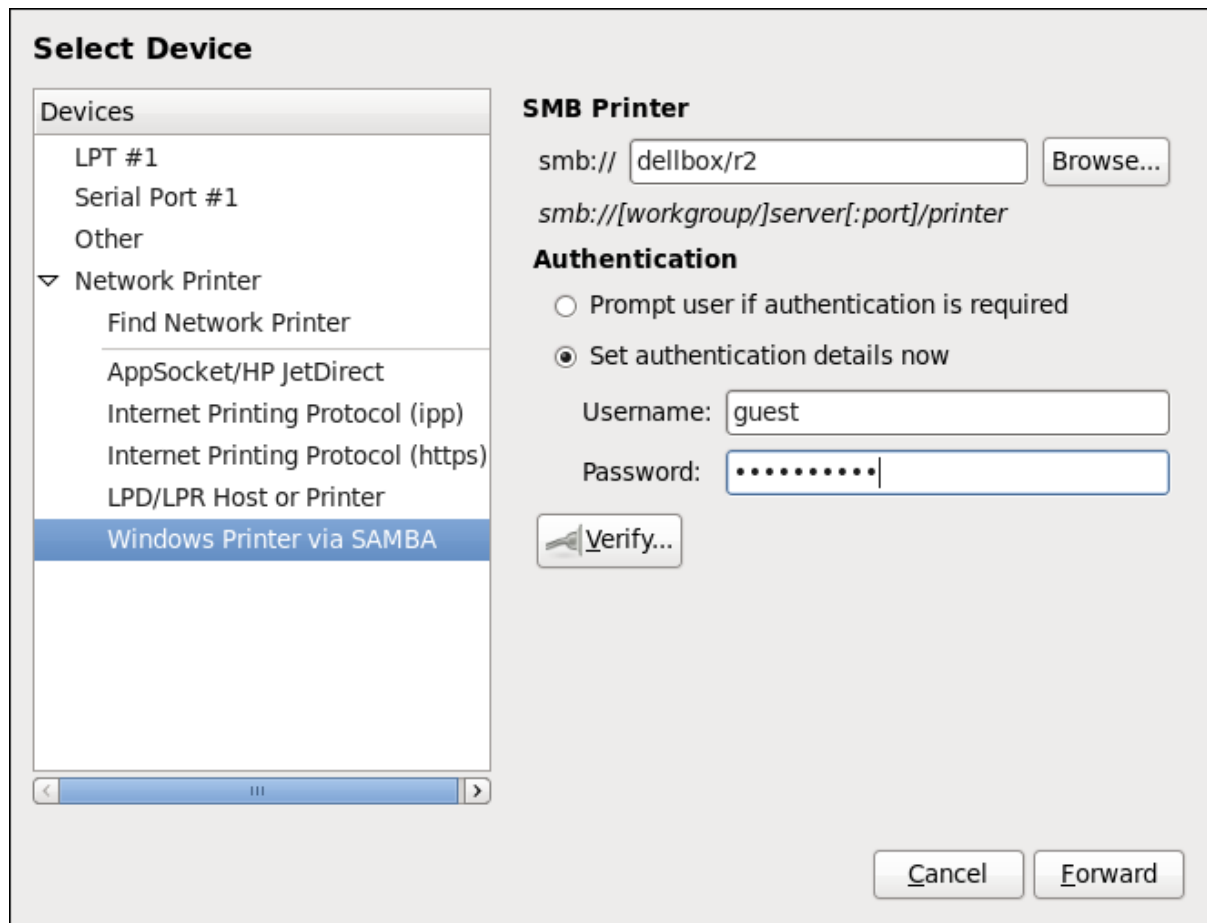
Note that in order to add a Samba printer, you need to have the samba-client package installed. You can do so by running, as **root**:

```
yum install samba-client
```

For more information on installing packages with Yum, see [Section 8.2.4, “Installing Packages”](#).

1. Open the **New Printer** dialog (see [Section 21.3.2, “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 21.8, “Adding a SMB printer”, the *computer name* is **dellbox** and the *printer share* is **r2**.



**Figure 21.8. 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:
  - o **Prompt user if authentication is required**: user name and password are collected from the user when printing a document.
  - o **Set authentication details now**: provide authentication information now so it is not required later. In the **Username** field, enter the user name 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.



## WARNING

Samba printer user names and passwords are stored in the printer server as unencrypted files readable by root and the Linux Printing Daemon, lpd. Thus, other users that have root access to the printer server can view the user name and password you use to access the Samba printer.

Therefore, when you choose a user name 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.

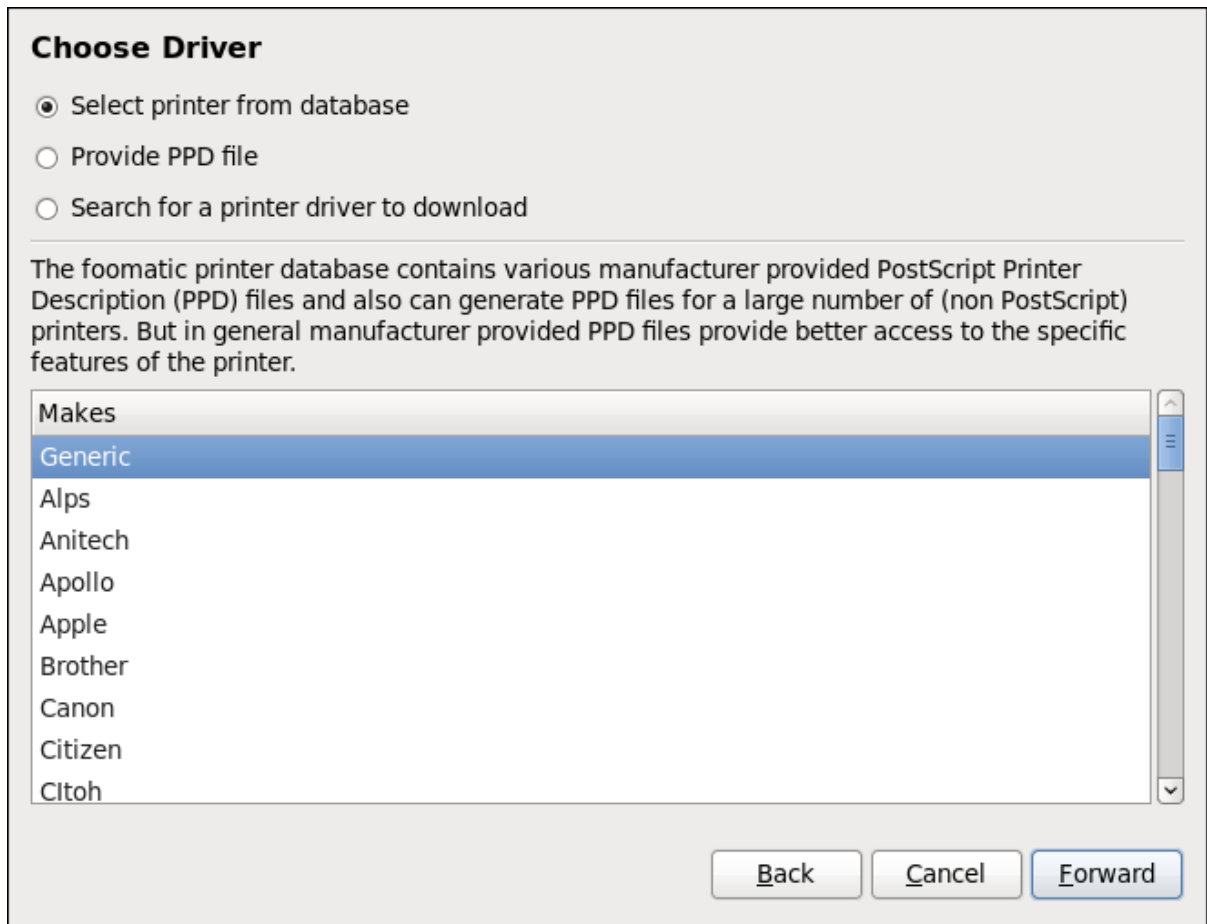
7. Click **Verify** to test the connection. Upon successful verification, a dialog box appears confirming printer share accessibility.
8. Click **Forward**.
9. Select the printer model. See [Section 21.3.8, “Selecting the Printer Model and Finishing”](#) for details.

### 21.3.8. 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:
  - **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**.
  - **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.
  - **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.



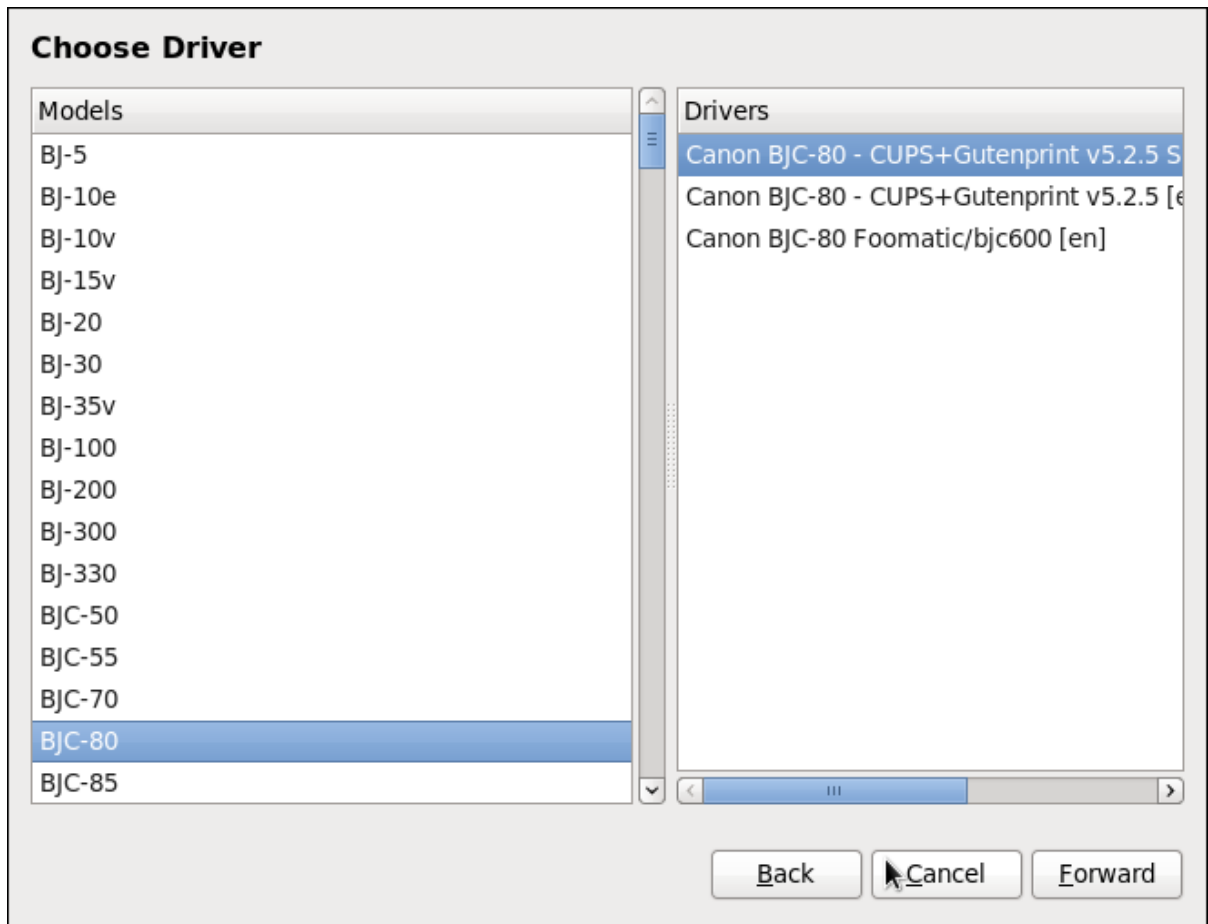
**Figure 21.9. Selecting a printer brand**

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 21.10](#), “[Selecting a printer model](#)” appears. Choose the corresponding model in the **Models** column on the left.



#### **NOTE**

On the right, the recommended printer 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 21.10. 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"

**Figure 21.11. 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 21.3.9, "Printing a Test Page"](#).

### 21.3.9. 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.

### 21.3.10. 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.

### 21.3.10.1. 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.

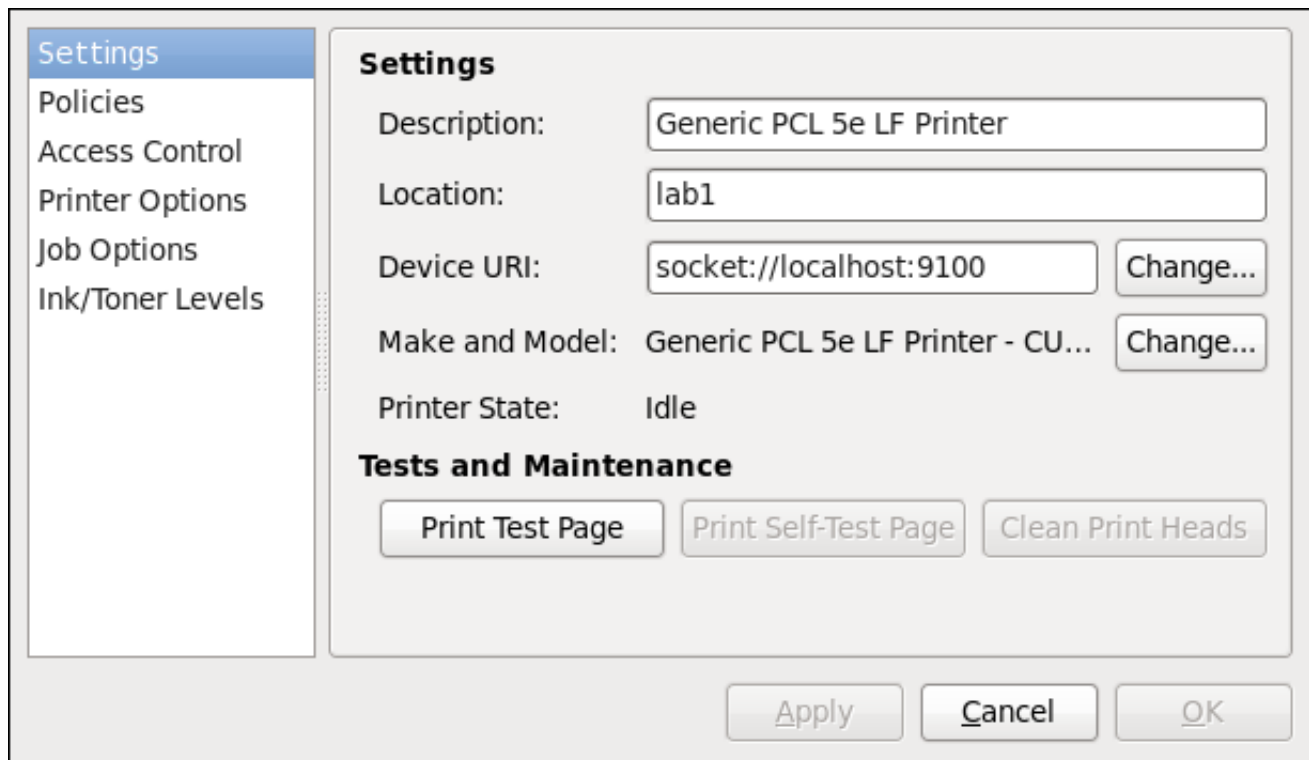


Figure 21.12. Settings page

### 21.3.10.2. 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 user name from the which the job originated, and the security status of the document being printed): click the **Starting Banner** or **Ending Banner** drop-down menu and choose the option that best describes the nature of the print jobs (for example, **confidential**).

#### 21.3.10.2.1. 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).

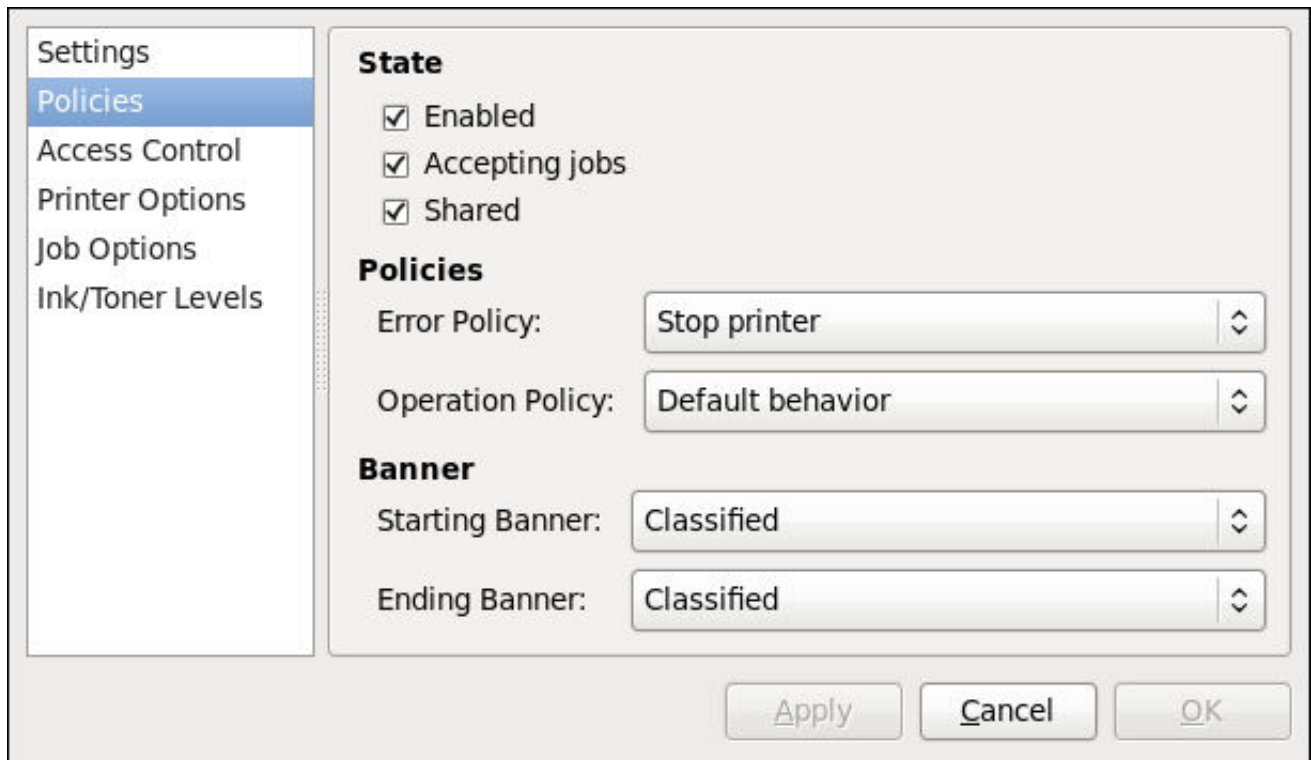


Figure 21.13. Policies page

#### 21.3.10.2.2. 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.

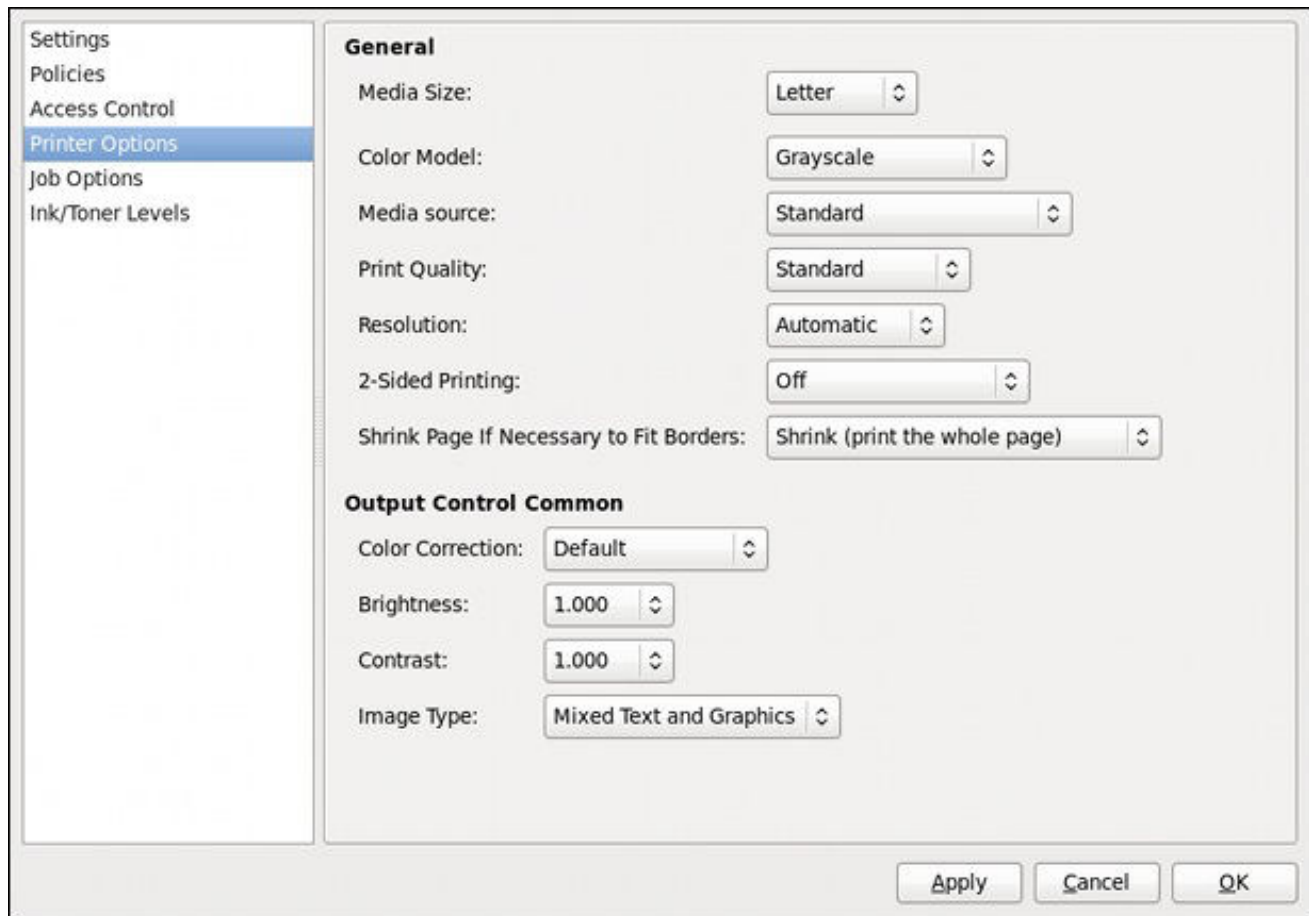


Figure 21.14. Access Control page



### 21.3.10.2.3. 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.



The screenshot displays the 'Printer Options' dialog box. On the left is a navigation pane with the following items: Settings, Policies, Access Control, Printer Options (highlighted), Job Options, and Ink/Toner Levels. The main area is divided into two sections:

- General**:
  - Media Size: Letter
  - Color Model: Grayscale
  - Media source: Standard
  - Print Quality: Standard
  - Resolution: Automatic
  - 2-Sided Printing: Off
  - Shrink Page If Necessary to Fit Borders: Shrink (print the whole page)
- Output Control Common**:
  - Color Correction: Default
  - Brightness: 1.000
  - Contrast: 1.000
  - Image Type: Mixed Text and Graphics

At the bottom right, there are three buttons: Apply, Cancel, and OK.

Figure 21.15. Printer Options page

### 21.3.10.2.4. 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.

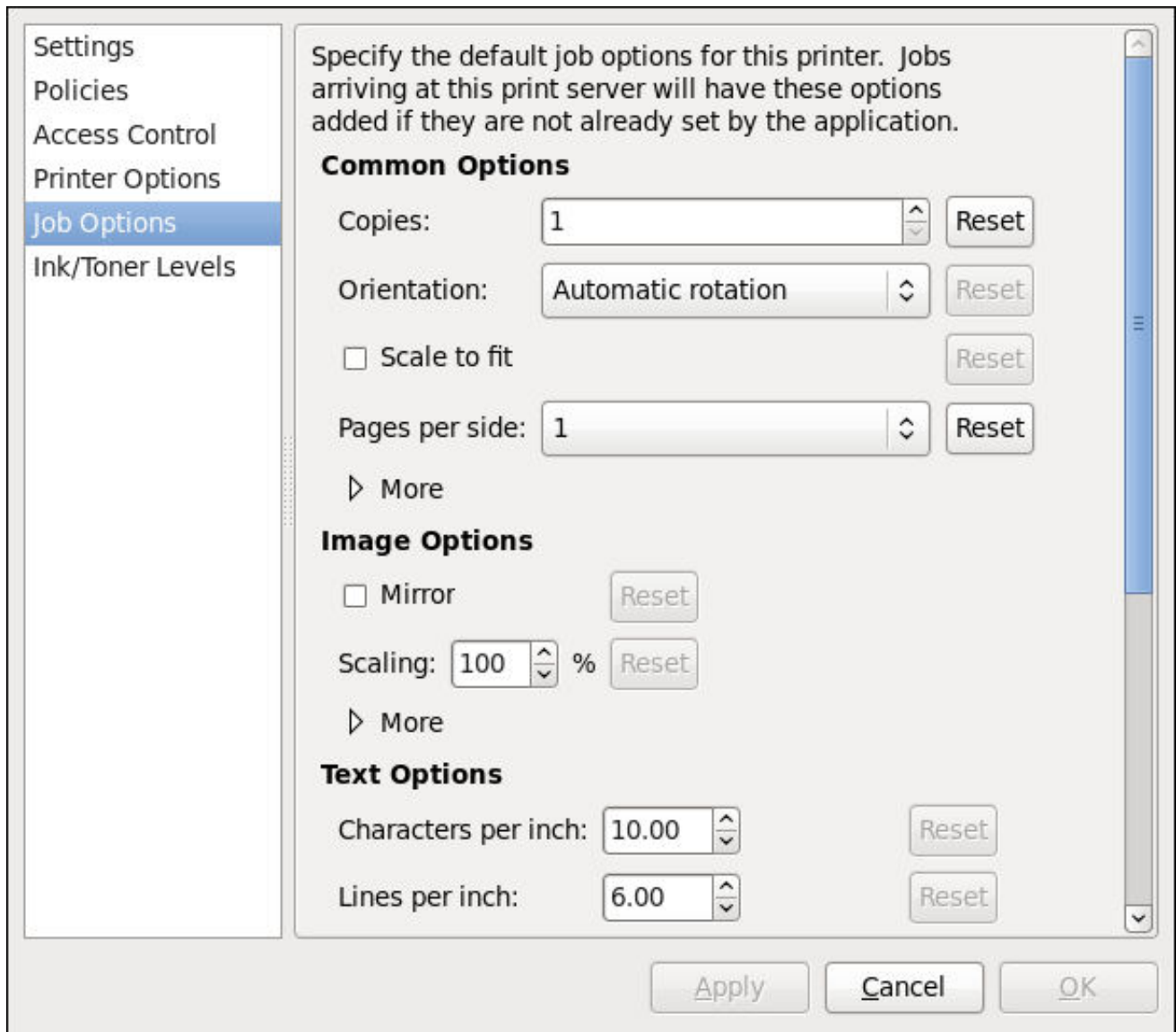


Figure 21.16. Job Options page

#### 21.3.10.2.5. 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.

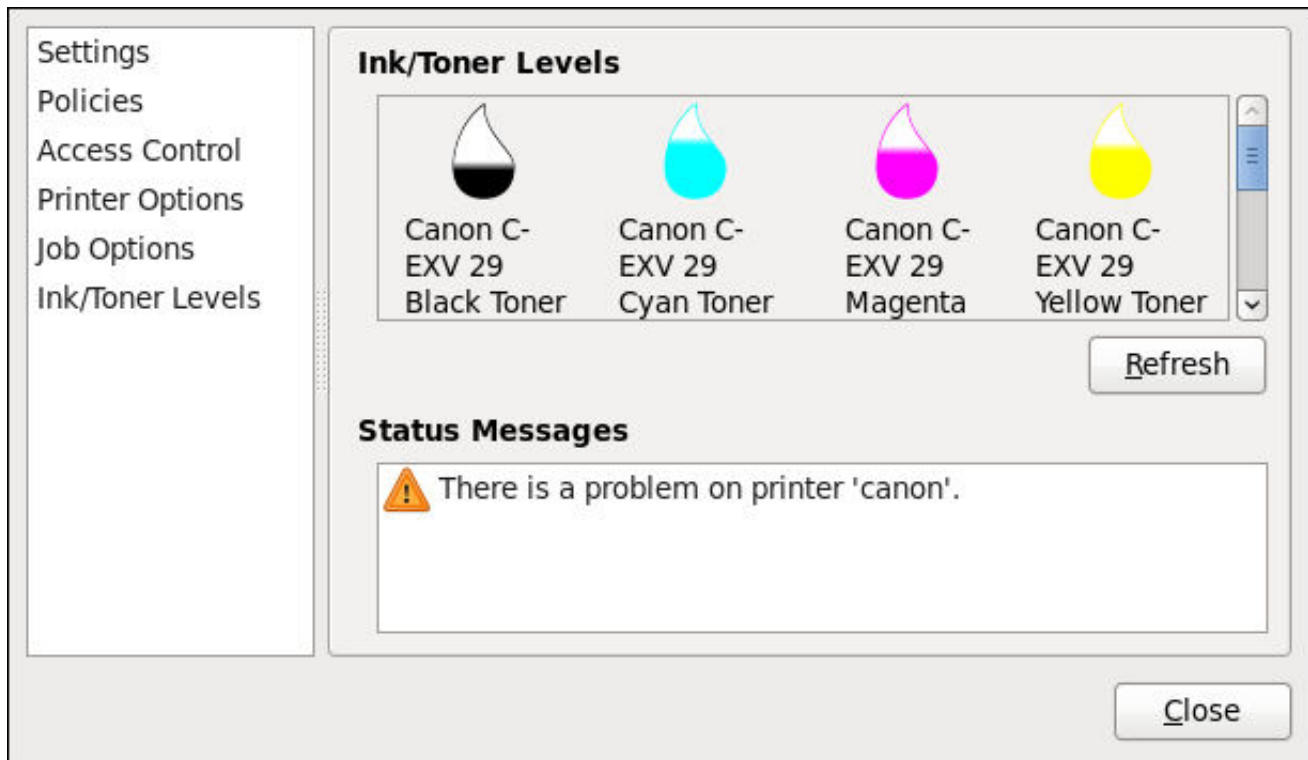


Figure 21.17. Ink/Toner Levels page

### 21.3.10.3. 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 21.18, “GNOME Print Status”](#).

File	Job	View				
Job	Document	Printer	Size	Time submitted	Status	
2	Red Hat	Generic-PCL-5e-LF-...	5k	a minute ago	Processing - Printer w...	
1	Product Document...	Canon	3k	22 hours ago	Pending	

Printer 'Generic-PCL-5e-LF-Printer': 'com.apple.print.recoverable'.

Figure 21.18. 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 21.11. 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 21.11](#), “[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.

**21.3.11. Additional Resources**

To learn more about printing on Red Hat Enterprise Linux, see the following resources.

**21.3.11.1. Installed Documentation****man lp**

The manual page for the **lp** command that allows you to print files from the command line.

**man lpr**

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.

### **21.3.11.2. 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 22. CONFIGURING NTP USING NTPD

### 22.1. 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 found at [www.pool.ntp.org](http://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](http://www.ntp.org) for more information. In Linux systems, **NTP** is implemented by a daemon running in user space. The default **NTP** daemon in Red Hat Enterprise Linux 6 is **ntpd**.

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.

### 22.2. 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.

## 22.3. 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 time stamps. The first 32-bits of the time stamp is used for seconds, the last 32-bits are used for fractions of 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 time stamps so this does not present the same level of problem as other implementations of time protocols have done. If a hardware clock that is within 68 years of the correct time 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 time stamps 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 time stamp”. When the packet arrives, the time server adds a “receive time stamp”. After processing the request for time and date information and just before returning the packet, it adds a “transmit time stamp”. When the returning packet arrives at the **NTP** client, a “receive time stamp” 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.

When a packet containing time information is received it is not immediately responded to, but is first subject to validation checks and then processed together with several other time samples to arrive at an estimate of the time. This is then compared to the system clock to determine the time offset, 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, Date and Time Configuration](#). 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/ntpd` configuration file.

## 22.4. 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 **ntpd** has write permissions.

## 22.5. 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 check box **System Clock Uses UTC** in the **system-config-date** graphical configuration tool. See [Chapter 2, Date and Time Configuration](#) 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 22.19, “Additional Resources”](#).

## 22.6. 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 22.16.12, “Configuring Symmetric Authentication Using a Key”](#) for an explanation of the **key** option.

## 22.7. 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*.

## 22.8. 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.

## 22.9. 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 22.16, “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 lines setup the default access control restrictions:

```
restrict default kod nomodify notrap nopeer noquery
restrict -6 default kod nomodify notrap nopeer noquery
```

The `kod` option means a “Kiss-o'-death” packet is to be sent to reduce unwanted queries. The `nomodify` options prevents any changes to the configuration. The `notrap` option prevents `ntpd` control message protocol traps. The `nopeer` option prevents a peer association being formed. The `noquery` option prevents `ntpq` and `ntpd` queries, but not time queries, from being answered. The `-6` option is required before an **IPv6** address.

Addresses within the range `127.0.0.0/8` 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 -6 ::1
```

Addresses can be added underneath if specifically required by another application. The `-6` option is required before an **IPv6** address.

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 `192.0.2.250/32`, 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

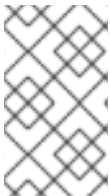
By default, as of Red Hat Enterprise 6.5, the **ntp.conf** file contains four public server entries:

```
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
```

If upgrading from a previous minor release, and your **/etc/ntp.conf** file has been modified, then the upgrade to Red Hat Enterprise Linux 6.5 will create a new file **/etc/ntp.conf.rpmnew** and will not alter the existing **/etc/ntp.conf** file.

### 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 22.16, “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**.

## 22.10. UNDERSTANDING THE NTPD SYSCONFIG FILE

The file will be read by the **ntpd** init script on service start. The default contents is as follows:

```
# Drop root to id 'ntp:ntp' by default.
OPTIONS="-u ntp:ntp -p /var/run/ntpd.pid -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.

The **-p** option sets the path to the pid file and **-u** sets the user and group to which the daemon should drop the root privileges.

## 22.11. 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.

## 22.12. INSTALLING THE NTP DAEMON (NTPD)

To install **ntpd**, enter the following command as **root**:

```
~]# yum install ntp
```

The default installation directory is **/usr/sbin/**.

## 22.13. CHECKING THE STATUS OF NTP

To check if **ntpd** is configured to run at system start, issue the following command:

```
~]$ chkconfig --list ntpd
ntpd          0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

By default, when **ntpd** is installed, it is configured to start at every system start.

To check if **ntpd** is running, issue the following command:

```
~]$ ntpq -p
      remote               refid           st t when poll reach  delay  offset
jitter
=====
====
+clock.util.phx2 .CDMA.          1 u  111  128  377  175.495  3.076
2.250
*clock02.util.ph .CDMA.          1 u   69  128  377  175.357  7.641
3.671
  ms21.snowflakeh .STEP.         16 u   - 1024    0    0.000  0.000
0.000
  rs11.lvs.iif.hu .STEP.         16 u   - 1024    0    0.000  0.000
0.000
  2001:470:28:bde .STEP.         16 u   - 1024    0    0.000  0.000
0.000
```

The command lists connected time servers and displays information indicating when they were last polled and the stability of the replies. The column headings are as follows:

- remote and refid: remote NTP server, and its NTP server
- st: stratum of server
- t: type of server (local, unicast, multicast, or broadcast)
- poll: how frequently to query server (in seconds)
- when: how long since last poll (in seconds)
- reach: octal bitmask of success or failure of last 8 queries (left-shifted); 377 = 11111111 = all recent queries were successful; 257 = 10101111 = 4 most recent were successful, 5 and 7 failed
- delay: network round trip time (in milliseconds)
- offset: difference between local clock and remote clock (in milliseconds)

- jitter: difference of successive time values from server (high jitter could be due to an unstable clock or, more likely, poor network performance)

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
```

## 22.14. 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.

### 22.14.1. Configure the Firewall Using the Graphical Tool

To enable **NTP** to pass through the firewall, using the graphical tool **system-config-firewall**, issue the following command as root:

```
~]# system-config-firewall
```

The **Firewall Configuration** window opens. Select **Other Ports** from the list on the left. Click **Add**. The **Port and Protocol** window opens. Click on one of the port numbers and start typing **123**. Select the “port 123” entry with **udp** as the protocol. Click **OK**. The **Port and Protocol** window closes. Click **Apply** in the **Firewall Configuration** window to apply the changes. A dialog box will pop up to ask you to confirm the action, click **Yes**. Note that any existing sessions will be terminated when you click **Yes**.

### 22.14.2. Configure the Firewall Using the Command Line

To enable **NTP** to pass through the firewall using the command line, issue the following command as **root**:

```
~]# lokkit --port=123:udp --update
```

Note that this will restart the firewall as long as it has not been disabled with the **--disabled** option. Active connections will be terminated and time out on the initiating machine.

When preparing a configuration file for multiple installations using administration tools, it is useful to edit the firewall configuration file directly. Note that any mistakes in the configuration file could have unexpected consequences, cause an error, and prevent the firewall setting from being applied. Therefore, check the **/etc/sysconfig/system-config-firewall** file thoroughly after editing.

To enable **NTP** to pass through the firewall, by editing the configuration file, become the **root** user and add the following line to **/etc/sysconfig/system-config-firewall**:

```
--port=123:udp
```

Note that these changes will not take effect until the firewall is reloaded or the system restarted.

### 22.14.2.1. Checking Network Access for Incoming NTP Using the Command Line

To check if the firewall is configured to allow incoming **NTP** traffic for clients using the command line, issue the following command as **root**:

```
~]# less /etc/sysconfig/system-config-firewall
# Configuration file for system-config-firewall

--enabled
--service=ssh
```

In this example taken from a default installation, the firewall is enabled but **NTP** has not been allowed to pass through. Once it is enabled, the following line appears as output in addition to the lines shown above:

```
--port=123:udp
```

To check if the firewall is currently allowing incoming **NTP** traffic for clients, issue the following command as **root**:

```
~]# iptables -L -n | grep 'udp.*123'
ACCEPT      udp -- 0.0.0.0/0                0.0.0.0/0                state NEW
udp dpt:123
```

## 22.15. CONFIGURE NTPDATE SERVERS

The purpose of the **ntpd** service is to set the clock during system boot. This can be used to ensure that the services started after **ntpd** will have the correct time and will not observe a jump in the clock. The use of **ntpd** and the list of step-tickers is considered deprecated and so Red Hat Enterprise Linux 6 uses the **-g** option to the **ntpd** command by default and not **ntpd**. However, the **-g** option only enables **ntpd** to ignore the offset limit of 1000s and attempt to synchronize the time. It does not guarantee the time will be correct when other programs or services are started. Therefore the **ntpd** service can be useful when **ntpd** is disabled or if there are services which need to be started with the correct time and not observe a jump in the clock.

To check if the **ntpd** service is enabled to run at system start, issue the following command:

```
~]$ chkconfig --list ntpdate
ntpdate          0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

To enable the service to run at system start, issue the following command as **root**:

```
~]# chkconfig ntpdate on
```

To configure **ntpd** servers, using a text editor running as **root**, edit **/etc/ntp/step-tickers** to include one or more host names as follows:

```
clock1.example.com  
clock2.example.com
```

The number of servers listed is not very important as **ntpd** 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?

The **ntpd** service has a file that must contain a list of **NTP** servers to be used on system start. It is recommended to have at least four servers listed to reduce the chance of a “false ticker” (incorrect time source) influencing the quality of the time offset calculation. However, publicly accessible time sources are rarely incorrect.

## 22.16. 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.

### 22.16.1. 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 **ntpd** 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 **ntpd** queries are not affected. For more information on the **discard** command and the default values, see [Section 22.16.2, “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 **ntpd** queries, but not time queries, from being answered.

- **nopeer** — prevents a peer association being formed.
- **noserve** — deny all packets except **ntpq** and **ntpd** queries.
- **notrap** — prevents **ntpd** 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 **ntpd** should reply with a **KoD** packet, the **restrict** command needs to have both **limited** and **kod** options.

The **ntpq** and **ntpd** 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.

### 22.16.2. 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 22.16.1, “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
```

### 22.16.3. 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.

#### 22.16.4. 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.

#### 22.16.5. 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 22.6, “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 **FF05::101** (site local) to **NTP**. Administratively scoped **IPv4** multicast addresses can also be used, as described in [RFC 2365 Administratively Scoped IP Multicast](#).

#### 22.16.6. 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.

### 22.16.7. 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 22.6, “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.

### 22.16.8. Adding a Multicast Server Address

To add a multicast 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 **multicastserver** command in the **ntp.conf** file.

The **multicastserver** command takes the following form:

```
multicastserver 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.

### 22.16.9. 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.

### 22.16.10. 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, when the server responds, the system will send a burst of up to eight packets instead of the usual one packet. For use with the **server** command to improve the average quality of the time-offset calculations.

### 22.16.11. Configuring the **iburst** Option

To improve the time taken for initial synchronization, add the following option to the end of a server command:

**iburst**

When the server is unreachable, send a burst of eight packets instead of the usual one packet. The packet spacing is normally 2 s; however, the spacing between the first and second packets can be changed with the **calldelay** command to allow additional time for a modem or ISDN call to complete. For use with the **server** command to reduce the time taken for initial synchronization. As of Red Hat Enterprise Linux 6.5, this is now a default option in the configuration file.

### 22.16.12. 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 **manycastclient** 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
manycastclient 239.255.254.254 key 30
```

See also [Section 22.6, “Authentication Options for NTP”](#).

### 22.16.13. 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.

#### 22.16.14. 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.

#### 22.16.15. 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**.

#### 22.16.16. 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**.

### 22.17. 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.

### 22.18. 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.

## 22.19. ADDITIONAL RESOURCES

The following sources of information provide additional resources regarding **NTP** and **ntpd**.

### 22.19.1. 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.
- **ntpd(8)** man page — Describes the **ntpd** 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.

### 22.19.2. 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/manyopt.html>

Information on Automatic Server Discovery in **NTPv4**.

## CHAPTER 23. CONFIGURING PTP USING PTP4L

### 23.1. 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 6 now 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).

#### 23.1.1. 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.

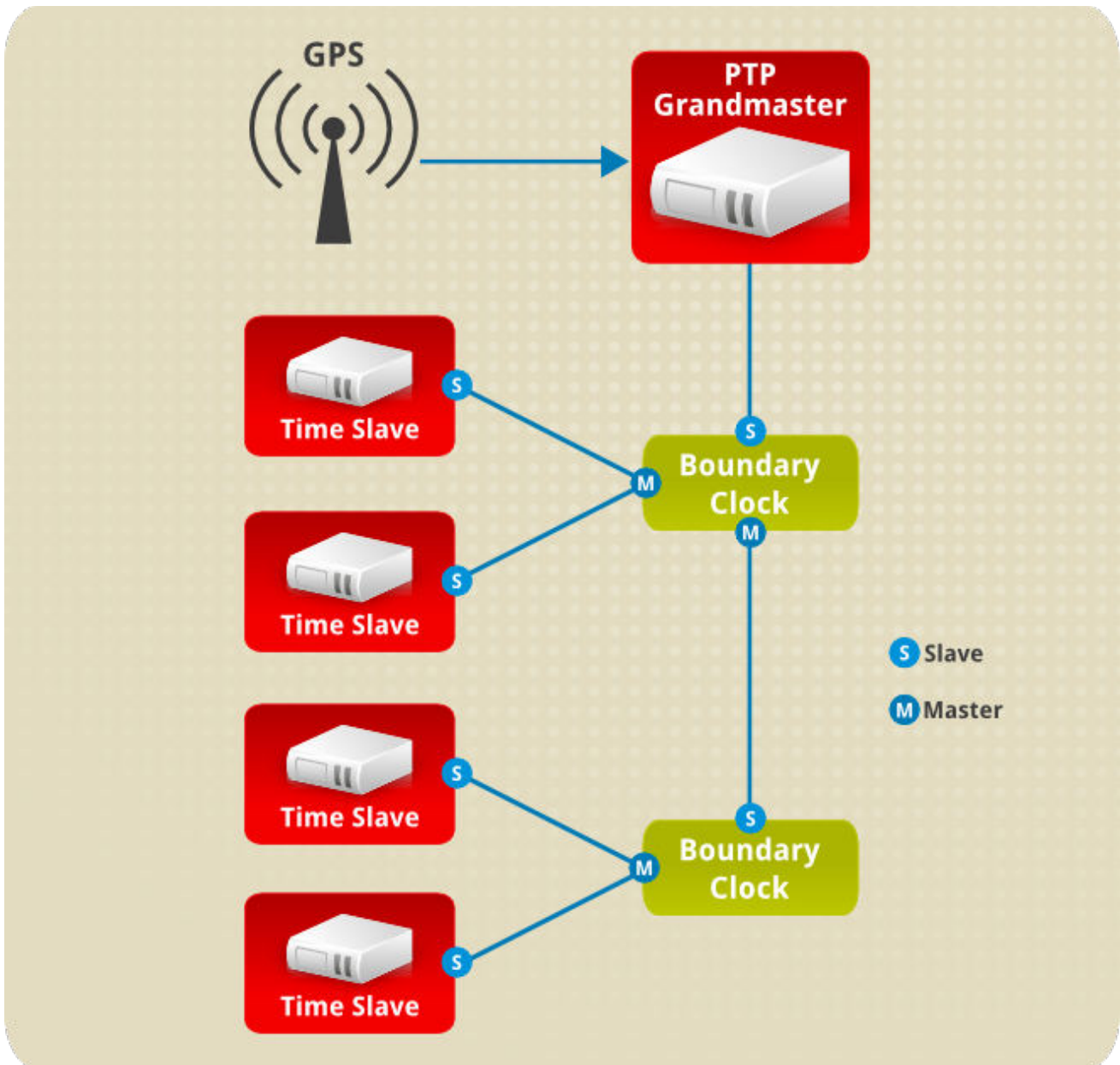


Figure 23.1. PTP grandmaster, boundary, and slave Clocks

### 23.1.2. 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.

## 23.2. 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.

### 23.2.1. 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 want 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**

### 23.2.2. Installing PTP

The kernel in Red Hat Enterprise Linux 6 now 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 23.7, "Serving PTP Time With NTP"](#).

### 23.2.3. Starting ptp4l

The **ptp4l** program tries to use hardware time stamping by default. To use **ptp4l** 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**:

```
~]# ptp4l -i eth3 -m
```

Where *eth3* is the interface you want to configure. Below is example output from **ptp4l** when the **PTP** clock on the NIC is synchronized to a master:

```
~]# ptp4l -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.ffff.0b552d-1
selected best master clock 00a069.ffff.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 **ptp4l(8)** man page). The **freq** 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 **ptp4l** program can also be started as a service by running:

```
~]# service ptp4l start
```

When running as a service, options are specified in the **/etc/sysconfig/ptp4l** file. More information on the different **ptp4l** options and the configuration file settings can be found in the **ptp4l(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:

```
~]# ptp4l -i eth3 -m -S
```

### 23.2.3.1. 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.

## 23.3. 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]

```

## 23.4. USING THE PTP MANAGEMENT CLIENT

The **PTP** management client, **pmc**, can be used to obtain additional information from **ptp4l** 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 **ptp4l** 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.

## 23.5. SYNCHRONIZING THE CLOCKS

The **phc2sys** program is used to synchronize the system clock to the **PTP** hardware clock (PHC) on the NIC. The **phc2sys** service is configured in the `/etc/sysconfig/phc2sys` configuration file. The default setting in the `/etc/sysconfig/phc2sys` file is as follows:

```
OPTIONS="-a -r"
```

The **-a** option causes **phc2sys** to read the clocks to be synchronized from the **ptp4l** application. It will follow changes in the **PTP** port states, adjusting the synchronization between the NIC hardware clocks accordingly. The system clock is not synchronized, unless the **-r** option is also specified. If you want the system clock to be eligible to become a time source, specify the **-r** option twice.

After making changes to `/etc/sysconfig/phc2sys`, restart the **phc2sys** service from the command line by issuing a command as **root**:

```
~]# service phc2sys restart
```

Under normal circumstances, use **service** commands to start, stop, and restart the **phc2sys** service.

When you do not want to start **phc2sys** as a service, you can start it from the command line. For example, enter the following command as **root**:

```
~]# phc2sys -a -r
```

The **-a** option causes **phc2sys** to read the clocks to be synchronized from the **ptp4l** application. If you want the system clock to be eligible to become a time source, specify the **-r** option twice.

Alternately, use the **-s** option to synchronize the system clock to a specific interface's **PTP** hardware clock. For example:

```
~]# phc2sys -s eth3 -w
```

The **-w** option waits for the running **ptp4l** application to synchronize the **PTP** clock and then retrieves the TAI to UTC offset from **ptp4l**.

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 36 seconds. The offset changes when leap seconds are inserted or deleted, which typically happens every few years. The **-0** option needs to be used to set this offset manually when the **-w** is not used, as follows:

```
~]# phc2sys -s eth3 -0 -36
```

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:

```
~]# service phc2sys start
```

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.

## 23.6. 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 `ptp4l` 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 **ptp4l** 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/ptp4l.conf** file:

```
summary_interval 10
```

An example of the **ptp4l** output, with **summary\_interval 6**, follows:

```

ptp4l: [615.253] selected /dev/ptp0 as PTP clock
ptp4l: [615.255] port 1: INITIALIZING to LISTENING on INITIALIZE
ptp4l: [615.255] port 0: INITIALIZING to LISTENING on INITIALIZE
ptp4l: [615.564] port 1: new foreign master 00a069.ffffe.0b552d-1
ptp4l: [619.574] selected best master clock 00a069.ffffe.0b552d
ptp4l: [619.574] port 1: LISTENING to UNCALIBRATED on RS_SLAVE
ptp4l: [623.573] port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l: [684.649] rms 669 max 3691 freq -29383 ± 3735 delay 9232 ± 122
ptp4l: [748.724] rms 253 max 588 freq -29787 ± 221 delay 9219 ± 158
ptp4l: [812.793] rms 287 max 673 freq -29802 ± 248 delay 9211 ± 183
ptp4l: [876.853] rms 226 max 534 freq -29795 ± 197 delay 9221 ± 138
ptp4l: [940.925] rms 250 max 562 freq -29801 ± 218 delay 9199 ± 148
ptp4l: [1004.988] rms 226 max 525 freq -29802 ± 196 delay 9228 ± 143
ptp4l: [1069.065] rms 300 max 646 freq -29802 ± 259 delay 9214 ± 176
ptp4l: [1133.125] rms 226 max 505 freq -29792 ± 197 delay 9225 ± 159
ptp4l: [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
```

## 23.7. 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**.

## 23.8. 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
```

## 23.9. SYNCHRONIZE TO PTP OR NTP TIME USING TIMEMASTER

When there are multiple **PTP** domains available on the network, or fallback to **NTP** is needed, the **timemaster** program can be used to synchronize the system clock to all available time sources. The **PTP** time is provided by **phc2sys** and **ptp4l** via *shared memory driver* (SHM reference clocks to



**chronyd** or **ntpd** (depending on the **NTP** daemon that has been configured on the system). The **NTP** daemon can then compare all time sources, both **PTP** and **NTP**, and use the best sources to synchronize the system clock.

On start, **timemaster** reads a configuration file that specifies the **NTP** and **PTP** time sources, checks which network interfaces have their own or share a **PTP** hardware clock (PHC), generates configuration files for **ptp4l** and **chronyd** or **ntpd**, and starts the **ptp4l**, **phc2sys**, and **chronyd** or **ntpd** processes as needed. It will remove the generated configuration files on exit. It writes configuration files for **chronyd**, **ntpd**, and **ptp4l** to `/var/run/timemaster/`.

### 23.9.1. Starting timemaster as a Service

To start **timemaster** as a service, issue the following command as **root**:

```
~]# service timemaster start
```

This will read the options in `/etc/timemaster.conf`. For more information on managing system services in Red Hat Enterprise Linux 6, see [Managing Services with systemd](#).

### 23.9.2. Understanding the timemaster Configuration File

Red Hat Enterprise Linux provides a default `/etc/timemaster.conf` file with a number of sections containing default options. The section headings are enclosed in brackets.

To view the default configuration, issue a command as follows:

```
~]$ less /etc/timemaster.conf
# Configuration file for timemaster

#[ntp_server ntp-server.local]
#minpoll 4
#maxpoll 4

#[ptp_domain 0]
#interfaces eth0

[timemaster]
ntp_program chronyd

[chrony.conf]
include /etc/chrony.conf

[ntp.conf]
includefile /etc/ntp.conf

[ptp4l.conf]

[chronyd]
path /usr/sbin/chronyd
options -u chrony

[ntpd]
path /usr/sbin/ntpd
options -u ntp:ntp -g
```

```
[phc2sys]
path /usr/sbin/phc2sys

[ptp4l]
path /usr/sbin/ptp4l
```

Notice the section named as follows:

```
[ntp_server address]
```

This is an example of an **NTP** server section, “ntp-server.local” is an example of a host name for an **NTP** server on the local LAN. Add more sections as required using a host name or **IP** address as part of the section name. Note that the short polling values in that example section are not suitable for a public server, see [Chapter 22, Configuring NTP Using ntpd](#) for an explanation of suitable **minpoll** and **maxpoll** values.

Notice the section named as follows:

```
[ptp_domain number]
```

A “PTP domain” is a group of one or more **PTP** clocks that synchronize to each other. They may or may not be synchronized to clocks in another domain. Clocks that are configured with the same domain number make up the domain. This includes a **PTP** grandmaster clock. The domain number in each “PTP domain” section needs to correspond to one of the **PTP** domains configured on the network.

An instance of **ptp4l** is started for every interface which has its own **PTP** clock and hardware time stamping is enabled automatically. Interfaces that support hardware time stamping have a **PTP** clock (PHC) attached, however it is possible for a group of interfaces on a NIC to share a PHC. A separate **ptp4l** instance will be started for each group of interfaces sharing the same PHC and for each interface that supports only software time stamping. All **ptp4l** instances are configured to run as a slave. If an interface with hardware time stamping is specified in more than one **PTP** domain, then only the first **ptp4l** instance created will have hardware time stamping enabled.

Notice the section named as follows:

```
[timemaster]
```

The default **timemaster** configuration includes the system **ntpd** and **chrony** configuration (**/etc/ntp.conf** or **/etc/chronyd.conf**) in order to include the configuration of access restrictions and authentication keys. That means any **NTP** servers specified there will be used with **timemaster** too.

The section headings are as follows:

- **[ntp\_server ntp-server.local]** — Specify polling intervals for this server. Create additional sections as required. Include the host name or **IP** address in the section heading.
- **[ptp\_domain 0]** — Specify interfaces that have **PTP** clocks configured for this domain. Create additional sections with, the appropriate domain number, as required.
- **[timemaster]** — Specify the **NTP** daemon to be used. Possible values are **chronyd** and **ntpd**.
- **[chrony.conf]** — Specify any additional settings to be copied to the configuration file generated for **chronyd**.

- **[ntp.conf]** — Specify any additional settings to be copied to the configuration file generated for **ntpd**.
- **[ptp41.conf]** — Specify options to be copied to the configuration file generated for **ptp4l**.
- **[chronyd]** — Specify any additional settings to be passed on the command line to **chronyd**.
- **[ntpd]** — Specify any additional settings to be passed on the command line to **ntpd**.
- **[phc2sys]** — Specify any additional settings to be passed on the command line to **phc2sys**.
- **[ptp41]** — Specify any additional settings to be passed on the command line to all instances of **ptp4l**.

The section headings and their contents are explained in detail in the **timemaster(8)** manual page.

### 23.9.3. Configuring timemaster Options

#### Procedure 23.1. Editing the timemaster Configuration File

1. To change the default configuration, open the `/etc/timemaster.conf` file for editing as **root**:

```
~]# vi /etc/timemaster.conf
```

2. For each **NTP** server you want to control using **timemaster**, create **[ntp\_server address]** sections. Note that the short polling values in the example section are not suitable for a public server, see [Chapter 22, Configuring NTP Using ntpd](#) for an explanation of suitable **minpoll** and **maxpoll** values.
3. To add interfaces that should be used in a domain, edit the **#[ptp\_domain 0]** section and add the interfaces. Create additional domains as required. For example:

```
[ptp_domain 0]
    interfaces eth0

    [ptp_domain 1]
    interfaces eth1
```

4. If required to use **ntpd** as the **NTP** daemon on this system, change the default entry in the **[timemaster]** section from **chronyd** to **ntpd**. See [Configuring NTP Using the chrony Suite](#) for information on the differences between **ntpd** and **chronyd**.
5. If using **chronyd** as the **NTP** server on this system, add any additional options below the default **include /etc/chrony.conf** entry in the **[chrony.conf]** section. Edit the default **include** entry if the path to `/etc/chrony.conf` is known to have changed.
6. If using **ntpd** as the **NTP** server on this system, add any additional options below the default **include /etc/ntp.conf** entry in the **[ntp.conf]** section. Edit the default **include** entry if the path to `/etc/ntp.conf` is known to have changed.
7. In the **[ptp41.conf]** section, add any options to be copied to the configuration file generated for **ptp4l**. This chapter documents common options and more information is available in the **ptp4l(8)** manual page.

8. In the **[chronyd]** section, add any command line options to be passed to **chronyd** when called by **timemaster**. See *Configuring NTP Using the chrony Suite* for information on using **chronyd**.
9. In the **[ntpd]** section, add any command line options to be passed to **ntpd** when called by **timemaster**. See *Chapter 22, Configuring NTP Using ntpd* for information on using **ntpd**.
10. In the **[phc2sys]** section, add any command line options to be passed to **phc2sys** when called by **timemaster**. This chapter documents common options and more information is available in the **phy2sys(8)** manual page.
11. In the **[ptp4l]** section, add any command line options to be passed to **ptp4l** when called by **timemaster**. This chapter documents common options and more information is available in the **ptp4l(8)** manual page.
12. Save the configuration file and restart **timemaster** by issuing the following command as **root**:

```
~]# service timemaster restart
```

## 23.10. IMPROVING ACCURACY

Previously, test results indicated that disabling the tickless kernel capability could 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. However, recent improvements applied to **kernel-3.10.0-197.el7** have greatly improved the stability of the system clock and the difference in stability of the clock with and without **nohz=off** should be much smaller now for most users.

The **ptp4l** and **phc2sys** applications can be configured to use a new adaptive servo. The advantage over the PI servo is that it does not require configuration of the PI constants to perform well. To make use of this for **ptp4l**, add the following line to the **/etc/ptp4l.conf** file:

```
clock_servo linreg
```

After making changes to **/etc/ptp4l.conf**, restart the **ptp4l** service from the command line by issuing the following command as **root**:

```
~]# service ptp4l restart
```

To make use of this for **phc2sys**, add the following line to the **/etc/sysconfig/phc2sys** file:

```
-E linreg
```

After making changes to **/etc/sysconfig/phc2sys**, restart the **phc2sys** service from the command line by issuing the following command as **root**:

```
~]# service phc2sys restart
```

## 23.11. ADDITIONAL RESOURCES

The following sources of information provide additional resources regarding **PTP** and the **ptp4l** tools.

### 23.11.1. 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).

### 23.11.2. Useful Websites

<http://linuxptp.sourceforge.net/>

The Linux PTP project.

<http://www.nist.gov/el/isd/ieee/ieee1588.cfm>

The IEEE 1588 Standard.

## **PART VII. MONITORING AND AUTOMATION**

This part describes various tools that allow system administrators to monitor system performance, automate system tasks, and report bugs.

## CHAPTER 24. 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.

### 24.1. VIEWING SYSTEM PROCESSES

#### 24.1.1. 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  [ksoftirqd/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 user name 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  [ksoftirqd/0]
root           5  0.0  0.0      0      0 ?        S    Mar02    0:00  [migration/0]
```

```
[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.

## 24.1.2. 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 user name of the process owner (**USER**), the priority (**PR**), the nice value (**NI**), the amount of virtual memory the process uses (**VRT**), the amount of non-swapped physical memory the process uses (**RES**), the amount of shared memory the process uses (**SHR**), the process status field **S**, the percentage of the CPU (**%CPU**) and memory (**%MEM**) usage, the accumulated 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     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     0  S   0.0   0.0   0:00.01  kthreadd
    3 root      RT   0     0     0     0  S   0.0   0.0   0:00.00  migration/0
    4 root      20   0     0     0     0  S   0.0   0.0   0:00.02  ksoftirqd/0
    5 root      RT   0     0     0     0  S   0.0   0.0   0:00.00  migration/0
    6 root      RT   0     0     0     0  S   0.0   0.0   0:00.00  watchdog/0
    7 root      20   0     0     0     0  S   0.0   0.0   0:01.00  events/0
    8 root      20   0     0     0     0  S   0.0   0.0   0:00.00  cpuset
```



```

  9 root      20   0   0   0   0 S  0.0  0.0  0:00.00 khelper
 10 root      20   0   0   0   0 S  0.0  0.0  0:00.00 netns
 11 root      20   0   0   0   0 S  0.0  0.0  0:00.00 async/mgr
 12 root      20   0   0   0   0 S  0.0  0.0  0:00.00 pm
[output truncated]

```

Table 24.1, “Interactive top commands” contains useful interactive commands that you can use with **top**. For more information, see the **top(1)** manual page.

**Table 24.1. Interactive top commands**

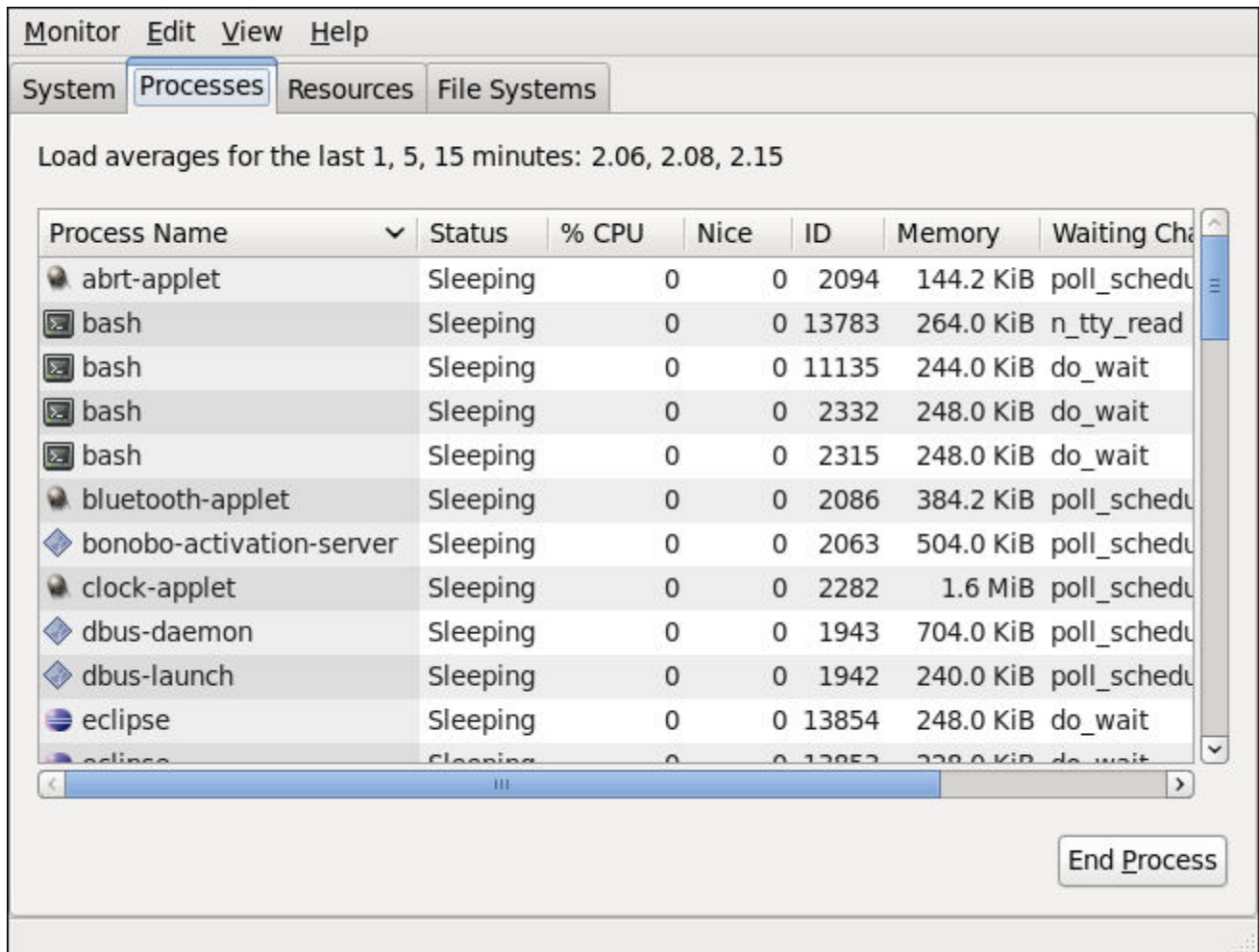
Command	Description
<b>Enter, Space</b>	Immediately refreshes the display.
<b>h, ?</b>	Displays a help screen.
<b>k</b>	Kills a process. You are prompted for the process ID and the signal to send to it.
<b>n</b>	Changes the number of displayed processes. You are prompted to enter the number.
<b>u</b>	Sorts the list by user.
<b>M</b>	Sorts the list by memory usage.
<b>P</b>	Sorts the list by CPU usage.
<b>q</b>	Terminates the utility and returns to the shell prompt.

### 24.1.3. 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 install the tool, issue the following command as **root**:

```
~]# yum install gnome-system-monitor
```

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 24.1. 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.

## 24.2. VIEWING MEMORY USAGE

### 24.2.1. 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.

## 24.2.2. 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.

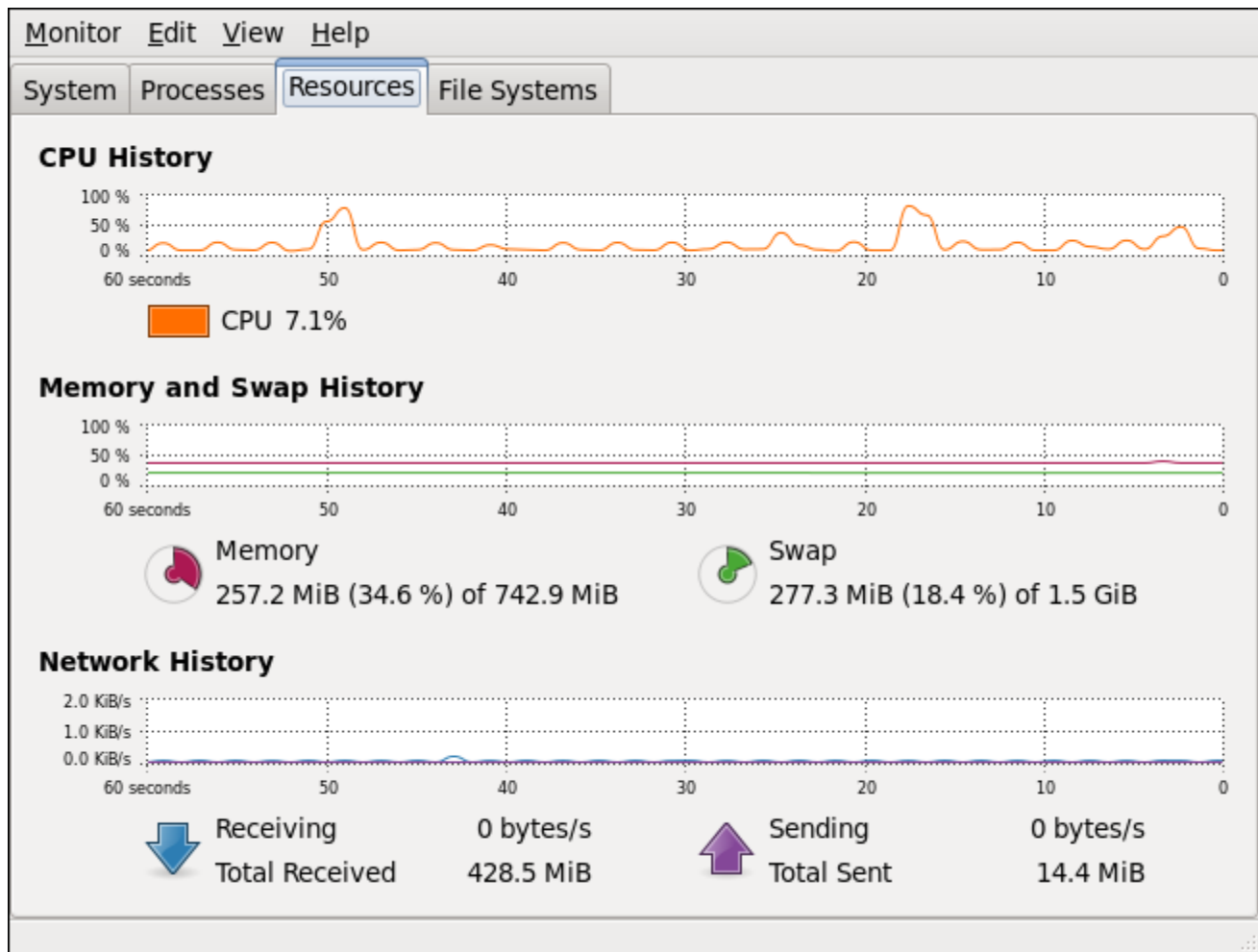


Figure 24.2. System Monitor — Resources

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.

## 24.3. VIEWING CPU USAGE

### 24.3.1. 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.

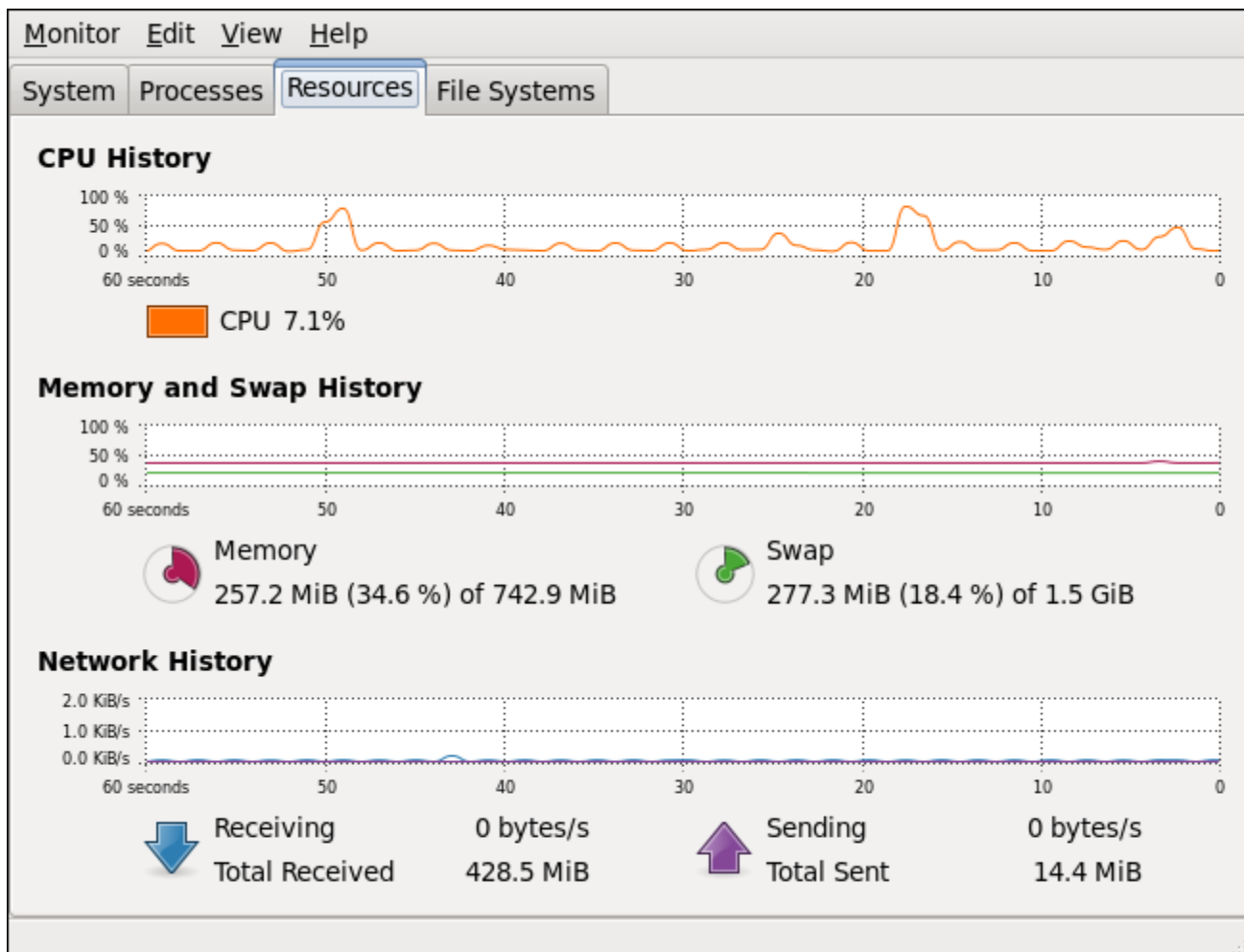


Figure 24.3. System Monitor — Resources

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.

## 24.4. VIEWING BLOCK DEVICES AND FILE SYSTEMS

### 24.4.1. 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.

## 24.4.2. 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 **blkid** 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.

### 24.4.3. 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
[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              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             rw,relatime
/var/lib/nfs/rpc_pipefs rpc_pipe             rw,relatime
/misc                 /etc/auto.misc        autofs
rw,relatime,fd=7,pg
/net                  -hosts               autofs
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 list all **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.

#### 24.4.4. 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.

**24.4.5. 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:

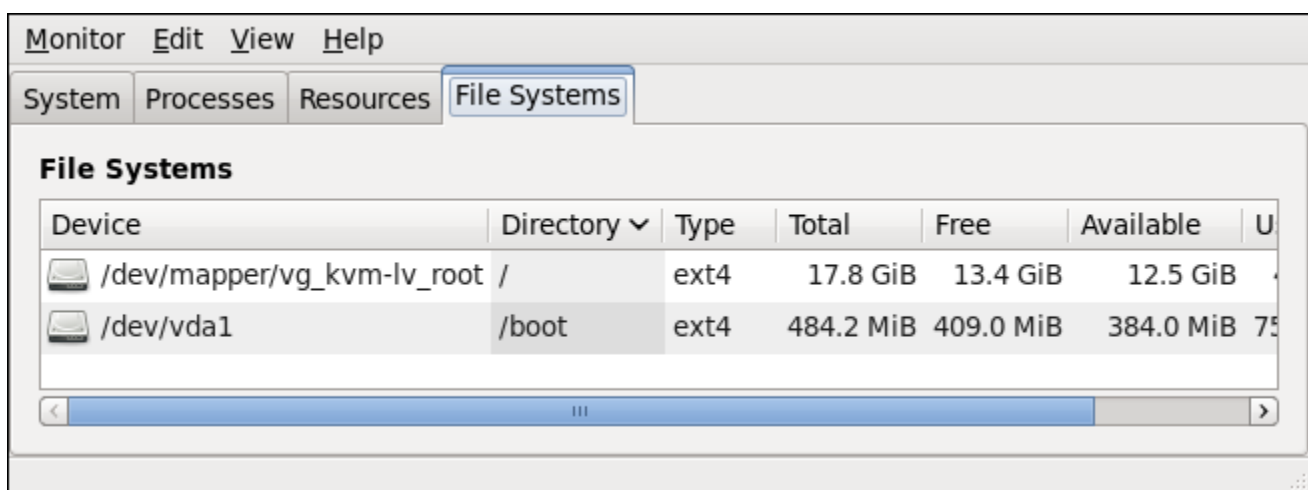
```
~]$ du -sh
15M    .
```

For a complete list of available command-line options, see the **du(1)** manual page.

### 24.4.6. 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 24.4. 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**).

### 24.4.7. Monitoring Files and Directories with gamin

Starting with Red Hat Enterprise Linux 6.8, the GLib system library uses **gamin** for monitoring of files and directories, and detection of their modifications on NFS file systems. By default, **gamin** uses polling for NFS file systems instead of **inotify**. Changes on other file systems are monitored by the inotify monitor that is implemented in GLib directly.

As a subset of the File Alteration Monitor (FAM) system, **gamin** re-implements the FAM specification with the inotify Linux kernel subsystem. It is a GNOME project, but without any GNOME dependencies. Both glib2 and gamin packages are installed by default.

By default, **gamin** works without the need of any configuration and it reverts to using polling for all paths matching `/mnt/*` or `/media/*` on Linux. Users can override or extend these settings by modifying the content of one of the following configuration files:

- `/etc/gamin/gaminrc`
- `$HOME/.gaminrc`
- `/etc/gamin/mandatory_gaminrc`

The configuration file accepts only the following commands:

### Commands accepted by the configuration file

#### **notify**

To express that kernel monitoring should be used for matching paths.

#### **poll**

To express that polling should be used for matching paths.

#### **fsset**

To control what notification method is used on a filesystem type.

An example of such configuration file can be seen here:

```
# configuration for gamin
# Can be used to override the default behaviour.
# notify filepath(s) : indicate to use kernel notification
# poll filepath(s)   : indicate to use polling instead
# fsset fsname method poll_limit : indicate what method of notification
for the file system
#
#                               kernel - use the kernel for
notification
#
#                               poll - use polling for notification
#                               none - don't use any notification
#                               the poll_limit is the number of
seconds
#
#                               that must pass before a resource is
polled again.
#
#                               It is optional, and if it is not
present the previous
#
#                               value will be used or the default.
notify /mnt/local* /mnt/pictures* # use kernel notification on these paths
```

```
poll /temp/*           # use poll notification on these paths
fsset nfs poll 10     # use polling on nfs mounts and poll
once every 10 seconds
```

The three configuration files are loaded in this order:

1. `/etc/gamin/gaminrc`
2. `~/.gaminrc`
3. `/etc/gamin/mandatory_gaminrc`

The `/etc/gamin/mandatory_gaminrc` configuration file allows the system administrator to override any potentially dangerous preferences set by the user. When checking a path to guess whether polling or kernel notification should be used, **gamin** checks first the user-provided rules in their declaration order within the configuration file and then check the predefined rules. This way the first declaration for `/mnt/local*` in the example override the default one for `/mnt/*`.

If **gamin** is not configured to use the poll notifications on a particular path, it decides based on the file system the path is located on.

## 24.5. VIEWING HARDWARE INFORMATION

### 24.5.1. 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.

## 24.5.2. 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 **lsusb(8)** manual page.

### 24.5.3. 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.

### 24.5.4. 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.

## 24.6. MONITORING PERFORMANCE WITH NET-SNMP

Red Hat Enterprise Linux 6 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.

### 24.6.1. 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 24.2, “Available Net-SNMP packages”](#) summarizes each of the packages and their contents.

**Table 24.2. Available Net-SNMP packages**

Package	Provides
net-snmp	The SNMP Agent Daemon and documentation. This package is required for exporting performance data.
net-snmp-libs	The <b>net snmp</b> library and the bundled management information bases (MIBs). This package is required for exporting performance data.
net-snmp-utils	SNMP clients such as <b>snmpget</b> and <b>snmpwalk</b> . This package is required in order to query a system's performance data over SNMP.
net-snmp-perl	The <b>mib2c</b> utility and the <b>NetSNMP</b> Perl module.

Package	Provides
net-snmp-python	An SNMP client library for Python.

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 8.2.4, “Installing Packages”](#).

## 24.6.2. 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 12, \*Services and Daemons\*](#).

### 24.6.2.1. Starting the Service

To run the `snmpd` service in the current session, type the following at a shell prompt as **root**:

```
service snmpd start
```

To configure the service to be automatically started at boot time, use the following command:

```
chkconfig snmpd on
```

This will enable the service in runlevel 2, 3, 4, and 5. Alternatively, you can use the **Service Configuration** utility as described in [Section 12.2.1.1, “Enabling and Disabling a Service”](#).

### 24.6.2.2. Stopping the Service

To stop the running `snmpd` service, type the following at a shell prompt as **root**:

```
service snmpd stop
```

To disable starting the service at boot time, use the following command:

```
chkconfig snmpd off
```

This will disable the service in all runlevels. Alternatively, you can use the **Service Configuration** utility as described in [Section 12.2.1.1, “Enabling and Disabling a Service”](#).



### 24.6.2.3. Restarting the Service

To restart the running **snmpd** service, type the following at a shell prompt:

```
service snmpd restart
```

This will stop the service and start it again in quick succession. To only reload the configuration without stopping the service, run the following command instead:

```
service snmpd reload
```

This will cause the running **snmpd** service to reload the configuration.

Alternatively, you can use the **Service Configuration** utility as described in [Section 12.2.1.2, “Starting, Restarting, and Stopping a Service”](#).

### 24.6.3. 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 6 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.



#### NOTE

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**:

```
service snmpd reload
```

#### 24.6.3.1. 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 host name. 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:

```
~]# service snmpd reload
Reloading snmpd: [ OK ]
~]# 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
```

### 24.6.3.2. 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”:

```
~]# service snmpd stop
Stopping snmpd: [ OK ]
~]# 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
~]# service snmpd start
Starting snmpd: [ OK ]
```

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 user name 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]
```

## 24.6.4. 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 24.6.3.2, “Configuring Authentication”](#).

### 24.6.4.1. 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 24.3, “Available OIDs”](#) summarizes the different OIDs available under that MIB.

**Table 24.3. Available OIDs**

OID	Description
<b>HOST-RESOURCES-MIB::hrSystem</b>	Contains general system information such as uptime, number of users, and number of running processes.
<b>HOST-RESOURCES-MIB::hrStorage</b>	Contains data on memory and file system usage.
<b>HOST-RESOURCES-MIB::hrDevices</b>	Contains a listing of all processors, network devices, and file systems.
<b>HOST-RESOURCES-MIB::hrSWRun</b>	Contains a listing of all running processes.
<b>HOST-RESOURCES-MIB::hrSWRunPerf</b>	Contains memory and CPU statistics on the process table from HOST-RESOURCES-MIB::hrSWRun.
<b>HOST-RESOURCES-MIB::hrSWInstalled</b>	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**:

```
~]$ snmptable -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.

#### 24.6.4.2. 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::ssIORawSent.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:

```
~]$ snmptable localhost UCD-SNMP-MIB::laTable
SNMP table: UCD-SNMP-MIB::laTable

 laIndex laNames laLoad laConfig laLoadInt laLoadFloat laErrorFlag
laErrorMessage
      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
```

#### 24.6.4.3. 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:

```
~]$ snmptable -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 **ssiORawRecieved.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:

```
~]$ snmptable -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

```

#### 24.6.4.4. 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:

```

~]$ snmptable -Cb localhost IF-MIB::ifTable
SNMP table: IF-MIB::ifTable

Index Descr          Type      Mtu    Speed      PhysAddress AdminStatus
   1    lo softwareLoopback 16436 10000000
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

```

#### 24.6.5. 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 that ship with 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 and Perl plug-ins. It assumes that the `net-snmp-utils` and `net-snmp-perl` packages are installed, and that the user is granted access to the SNMP tree as described in [Section 24.6.3.2, “Configuring Authentication”](#).

### 24.6.5.1. 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.



#### NOTE

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.

```

**WARNING**

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
```

**24.6.5.2. Extending Net-SNMP with Perl**

Executing shell scripts using the **extend** directive is a fairly limited method for exposing custom application metrics over SNMP. The Net-SNMP Agent also provides an embedded Perl interface for exposing custom objects. The `net-snmp-perl` package provides the **NetSNMP::agent** Perl module that is used to write embedded Perl plug-ins on Red Hat Enterprise Linux.

The **NetSNMP::agent** Perl module provides an **agent** object which is used to handle requests for a part of the agent's OID tree. The **agent** object's constructor has options for running the agent as a sub-agent of **snmpd** or a standalone agent. No arguments are necessary to create an embedded agent:

```
use NetSNMP::agent (':all');

my $agent = new NetSNMP::agent();
```

The **agent** object has a **register** method which is used to register a callback function with a particular OID. The **register** function takes a name, OID, and pointer to the callback function. The following example will register a callback function named **hello\_handler** with the SNMP Agent which will handle requests under the OID **.1.3.6.1.4.1.8072.9999.9999**:

```
$agent->register("hello_world", ".1.3.6.1.4.1.8072.9999.9999",
    \&hello_handler);
```

**NOTE**

The OID **.1.3.6.1.4.1.8072.9999.9999** (**NET-SNMP-MIB::netSnmPlaypen**) is typically used for demonstration purposes only. If your organization does not already have a root OID, you can obtain one by contacting an ISO Name Registration Authority (ANSI in the United States).

The handler function will be called with four parameters, **HANDLER**, **REGISTRATION\_INFO**,

**REQUEST\_INFO**, and **REQUESTS**. The **REQUESTS** parameter contains a list of requests in the current call and should be iterated over and populated with data. The **request** objects in the list have get and set methods which allow for manipulating the OID and value of the request. For example, the following call will set the value of a request object to the string "hello world":

```
$request->setValue(ASN_OCTET_STR, "hello world");
```

The handler function should respond to two types of SNMP requests: the GET request and the GETNEXT request. The type of request is determined by calling the **getMode** method on the **request\_info** object passed as the third parameter to the handler function. If the request is a GET request, the caller will expect the handler to set the value of the **request** object, depending on the OID of the request. If the request is a GETNEXT request, the caller will also expect the handler to set the OID of the request to the next available OID in the tree. This is illustrated in the following code example:

```
my $request;
my $string_value = "hello world";
my $integer_value = "8675309";

for($request = $requests; $request; $request = $request->next()) {
    my $oid = $request->getOID();
    if ($request_info->getMode() == MODE_GET) {
        if ($oid == new NetSNMP::OID(".1.3.6.1.4.1.8072.9999.9999.1.0")) {
            $request->setValue(ASN_OCTET_STR, $string_value);
        }
        elsif ($oid == new NetSNMP::OID(".1.3.6.1.4.1.8072.9999.9999.1.1")) {
            $request->setValue(ASN_INTEGER, $integer_value);
        }
    } elsif ($request_info->getMode() == MODE_GETNEXT) {
        if ($oid == new NetSNMP::OID(".1.3.6.1.4.1.8072.9999.9999.1.0")) {
            $request->setOID(".1.3.6.1.4.1.8072.9999.9999.1.1");
            $request->setValue(ASN_INTEGER, $integer_value);
        }
        elsif ($oid < new NetSNMP::OID(".1.3.6.1.4.1.8072.9999.9999.1.0")) {
            $request->setOID(".1.3.6.1.4.1.8072.9999.9999.1.0");
            $request->setValue(ASN_OCTET_STR, $string_value);
        }
    }
}
```

When **getMode** returns **MODE\_GET**, the handler analyzes the value of the **getOID** call on the **request** object. The value of the **request** is set to either **string\_value** if the OID ends in ".1.0", or set to **integer\_value** if the OID ends in ".1.1". If the **getMode** returns **MODE\_GETNEXT**, the handler determines whether the OID of the request is ".1.0", and then sets the OID and value for ".1.1". If the request is higher on the tree than ".1.0", the OID and value for ".1.0" is set. This in effect returns the "next" value in the tree so that a program like **snmpwalk** can traverse the tree without prior knowledge of the structure.

The type of the variable is set using constants from **NetSNMP::ASN**. See the **perldoc** for **NetSNMP::ASN** for a full list of available constants.

The entire code listing for this example Perl plug-in is as follows:

```
#!/usr/bin/perl

use NetSNMP::agent (':all');
```

```

use NetSNMP::ASN qw(ASN_OCTET_STR ASN_INTEGER);

sub hello_handler {
    my ($handler, $registration_info, $request_info, $requests) = @_;
    my $request;
    my $string_value = "hello world";
    my $integer_value = "8675309";

    for($request = $requests; $request; $request = $request->next()) {
        my $oid = $request->getOID();
        if ($request_info->getMode() == MODE_GET) {
            if ($oid == new NetSNMP::OID(".1.3.6.1.4.1.8072.9999.9999.1.0")) {
                $request->setValue(ASN_OCTET_STR, $string_value);
            }
            elsif ($oid == new NetSNMP::OID(".1.3.6.1.4.1.8072.9999.9999.1.1"))
        {
                $request->setValue(ASN_INTEGER, $integer_value);
            }
        } elsif ($request_info->getMode() == MODE_GETNEXT) {
            if ($oid == new NetSNMP::OID(".1.3.6.1.4.1.8072.9999.9999.1.0")) {
                $request->setOID(".1.3.6.1.4.1.8072.9999.9999.1.1");
                $request->setValue(ASN_INTEGER, $integer_value);
            }
            elsif ($oid < new NetSNMP::OID(".1.3.6.1.4.1.8072.9999.9999.1.0")) {
                $request->setOID(".1.3.6.1.4.1.8072.9999.9999.1.0");
                $request->setValue(ASN_OCTET_STR, $string_value);
            }
        }
    }
}

my $agent = new NetSNMP::agent();
$agent->register("hello_world", ".1.3.6.1.4.1.8072.9999.9999",
               \&hello_handler);

```

To test the plug-in, copy the above program to `/usr/share/snmp/hello_world.pl` and add the following line to the `/etc/snmp/snmpd.conf` configuration file:

```
perl do "/usr/share/snmp/hello_world.pl"
```

The SNMP Agent Daemon will need to be restarted to load the new Perl plug-in. Once it has been restarted, an `snmpwalk` should return the new data:

```

~]$ snmpwalk localhost NET-SNMP-MIB::netSnmPlaypen
NET-SNMP-MIB::netSnmPlaypen.1.0 = STRING: "hello world"
NET-SNMP-MIB::netSnmPlaypen.1.1 = INTEGER: 8675309

```

The `snmpget` should also be used to exercise the other mode of the handler:

```

~]$ snmpget localhost \
    NET-SNMP-MIB::netSnmPlaypen.1.0 \
    NET-SNMP-MIB::netSnmPlaypen.1.1
NET-SNMP-MIB::netSnmPlaypen.1.0 = STRING: "hello world"
NET-SNMP-MIB::netSnmPlaypen.1.1 = INTEGER: 8675309

```

## 24.7. ADDITIONAL RESOURCES

To learn more about gathering system information, see the following resources.

### 24.7.1. 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 25. 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 different log files for different 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 when trying to troubleshoot a problem with the system such as trying to load a kernel driver or when looking for unauthorized login attempts to the system. This chapter discusses where to find log files, how to view log files, and what to look for in log files.

Some log files are controlled by a daemon called **rsyslogd**. The **rsyslogd** daemon is an enhanced replacement for previous **sysklogd**, and provides extended filtering, encryption protected relaying of messages, various configuration options, input and output modules, support for transportation via the **TCP** or **UDP** protocols. Note that **rsyslog** is compatible with **sysklogd**.

### 25.1. INSTALLING RSYSLOG

Version 5 of **rsyslog**, provided in the **rsyslog** package, is installed by default in Red Hat Enterprise Linux 6. If required, to ensure that it is, issue the following command as **root**:

```
~]# yum install rsyslog
Loaded plugins: product-id, refresh-packagekit, subscription-manager
Package rsyslog-5.8.10-10.el6_6.i686 already installed and latest version
Nothing to do
```

#### 25.1.1. Upgrading to rsyslog version 7

Version 7 of **rsyslog**, provided in the **rsyslog7** package, is available in Red Hat Enterprise Linux 6. It provides a number of enhancements over version 5, in particular higher processing performance and support for more plug-ins. If required, to change to version 7, make use of the **yum shell** utility as described below.

##### Procedure 25.1. Upgrading to rsyslog 7

To upgrade from **rsyslog** version 5 to **rsyslog** version 7, it is necessary to install and remove the relevant packages simultaneously. This can be accomplished using the **yum shell** utility.

1. Enter the following command as **root** to start the yum shell:

```
~]# yum shell
Loaded plugins: product-id, refresh-packagekit, subscription-manager
>
```

The yum shell prompt appears.

2. Enter the following commands to install the **rsyslog7** package and remove the **rsyslog** package.

```
> install rsyslog7
> remove rsyslog
```

3. Enter **run** to start the process:

```
> run
```

```

--> Running transaction check
---> Package rsyslog.i686 0:5.8.10-10.el6_6 will be erased
---> Package rsyslog7.i686 0:7.4.10-3.el6_6 will be installed
--> Finished Dependency Resolution

=====
=====
Package           Arch      Version           Repository
Size
=====
=====
Installing:
 rsyslog7         i686      7.4.10-3.el6_6   rhel-6-workstation-rpms
1.3 M
Removing:
 rsyslog          i686      5.8.10-10.el6_6  @rhel-6-workstation-rpms
2.1 M

Transaction Summary
=====
=====
Install  1 Package
Remove  1 Package

Total download size: 1.3 M
Is this ok [y/d/N]:y

```

4. Enter **y** when prompted to start the upgrade.
5. When the upgrade is completed, the **yum shell** prompt is displayed. Enter **quit** or **exit** to exit the shell:

```

Finished Transaction
> quit
Leaving Shell
~]#

```

For information on using the new syntax provided by rsyslog version 7, see [Section 25.4, “Using the New Configuration Format”](#).

## 25.2. LOCATING LOG FILES

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. 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.

## 25.3. 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 the form of text following a hash sign (#).

### 25.3.1. 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 dot. 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 (or by a numerical code): **kern** (0), **user** (1), **mail** (2), **daemon** (3), **auth** (4), **syslog** (5), **lpr** (6), **news** (7), **uucp** (8), **cron** (9), **authpriv** (10), **ftp** (11), and **local0** through **local7** (16 - 23).
- *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 25.1. 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 [the section called “Properties”](#). You can compare each of the specified properties to a particular value using one of the compare-operations listed in [Table 25.1, “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 25.1, “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 25.1. Property-based compare-operations**

Compare-operation	Description
<i>contains</i>	Checks whether the provided string matches any part of the text provided by the property. To perform case-insensitive comparisons, use <i>contains_i</i> .
<i>isequal</i>	Compares the provided string against all of the text provided by the property. These two values must be exactly equal to match.
<i>startswith</i>	Checks whether the provided string is found exactly at the beginning of the text provided by the property. To perform case-insensitive comparisons, use <i>startswith_i</i> .

Compare-operation	Description
<i>regex</i>	Compares the provided POSIX BRE (Basic Regular Expression) against the text provided by the property.
<i>ereregex</i>	Compares the provided POSIX ERE (Extended Regular Expression) regular expression against the text provided by the property.
<i>isempty</i>	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 25.2. 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.

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 a single filter by using **and** 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 25.3, “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 25.3. 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")
}
```

See [the section called “Online Documentation”](#) for more examples of various expression-based filters. RainerScript is the basis for **rsyslog**'s new configuration format, see [Section 25.4, “Using the New Configuration Format”](#)

## 25.3.2. 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 improve 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 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 a colon (;). For more information on templates, see [the section called “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 25.4. 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 6514 and the **TCP** protocol, use:

```
*.* @@example.com:6514
```

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 25.3.5, “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 as an action inside a rule, type:

```
FILTER :omfile:$NAME
```

### Example 25.5. 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 want

to send the messages to (as in [Example 25.7, “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 25.3.3, “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 formatting template.

#### Example 25.6. 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
```



#### WARNING

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 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 **ommysql** 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 25.3.3, “Templates”](#).

## IMPORTANT

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 25.7, “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 25.7. 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 25.3.3, “Templates”](#).



#### WARNING

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*.

### 25.3.3. 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 between two percent signs (%) specifies a *property* that allows you to access specific contents of a syslog message. For more information on properties, see [the section called “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.





## NOTE

Note that the database writer 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.

See section *Storing syslog messages in a database* in [Section 25.3.2, “Actions”](#) for more information.

## 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 an action definition to mark the dynamic file name template:

```
*.* ?DynamicFile
```

## Properties

Properties defined inside a template (between two percent signs (%)) enable 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 a few examples of **rsyslog** templates.

[Example 25.8, “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 25.8. A verbose syslog message template

```
$template verbose, "%syslogseverity%, %syslogfacility%, %timegenerated%, %HOSTNAME%, %syslogtag%, %msg%\n"
```

[Example 25.9, “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 *mesg(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 25.9. A wall message template

```
$template wallmsg, "\r\n\7Message from syslogd@%HOSTNAME% at %timegenerated% ... \r\n %syslogtag% %msg%\n\r"
```

[Example 25.10, “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 25.10. 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's 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: '%rawmsg%'\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%\n"
```

### ***RSYSLOG\_TraditionalForwardFormat***

The traditional forwarding format with low-precision time stamps.

```
"%PRI%%TIMESTAMP% %HOSTNAME% %syslogtag:1:32%%msg:::sp-if-no-1st-
sp%%msg%\n"
```

## 25.3.4. 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 can 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 [the section called “Online Documentation”](#). Currently, a new configuration format has been developed that replaces the **\$**-based syntax (see [Section 25.4, “Using the New Configuration Format”](#)). However, classic global directives remain supported as a legacy format.

### 25.3.5. 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 rsyslogd
    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** — Specifies any options to be passed to the compression program used.
  - **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.

## 25.4. USING THE NEW CONFIGURATION FORMAT

In **rsyslog** version 7, available for Red Hat Enterprise Linux 6 in the **rsyslog7** package, a new configuration syntax is 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. To install **rsyslog** 7, see [Section 25.1.1, “Upgrading to rsyslog version 7”](#).

RainerScript is a scripting language designed for processing network events and configuring event processors such as **rsyslog**. The version of RainerScript in **rsyslog** version 5 is used to define expression-based filters, see [Example 25.3, “Expression-based Filters”](#). The version of RainerScript in **rsyslog** version 7 implements the **input()** and **ruleset()** statements, which permit the **/etc/rsyslog.conf** configuration file to be written in the new syntax. The new syntax differs mainly in that it is much more structured; parameters are passed as arguments to statements, such as **input**,

action, template, and module load. The scope of options is limited by blocks. This enhances readability and reduces the number of bugs caused by misconfiguration. There is also a significant performance gain. Some functionality is exposed in both syntaxes, some only in the new one.

Compare the configuration written with legacy-style parameters:

```
$InputFileName /tmp/inputfile
$InputFileTag tag1:
$InputFileStateFile inputfile-state
$InputRunFileMonitor
```

and the same configuration with the 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 [the section called “Online Documentation”](#).

### 25.4.1. 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 a 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. The legacy ruleset definition in `/etc/rsyslog.conf` can look as follows:

```
$RuleSet rulesetname
rule
rule2
```

The rule ends when another rule is defined, or the default ruleset is called as follows:

```
$RuleSet RSYSLOG_DefaultRuleset
```

With the new configuration format in rsyslog 7, the `input()` and `ruleset()` statements are reserved for this operation. The new format 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 [the section called “Online Documentation”](#).

### Example 25.11. 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-6514") {
    action(type="omfile" file="/var/log/remote-6514")
}

ruleset(name="remote-601") {
    cron.* action(type="omfile" file="/var/log/remote-601-cron")
    mail.* action(type="omfile" file="/var/log/remote-601-mail")
}

input(type="imtcp" port="6514" ruleset="remote-6514");
input(type="imtcp" port="601" ruleset="remote-601");
```

Rulesets shown in the above example define log destinations for the remote input from two ports, in case of port **601**, 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.

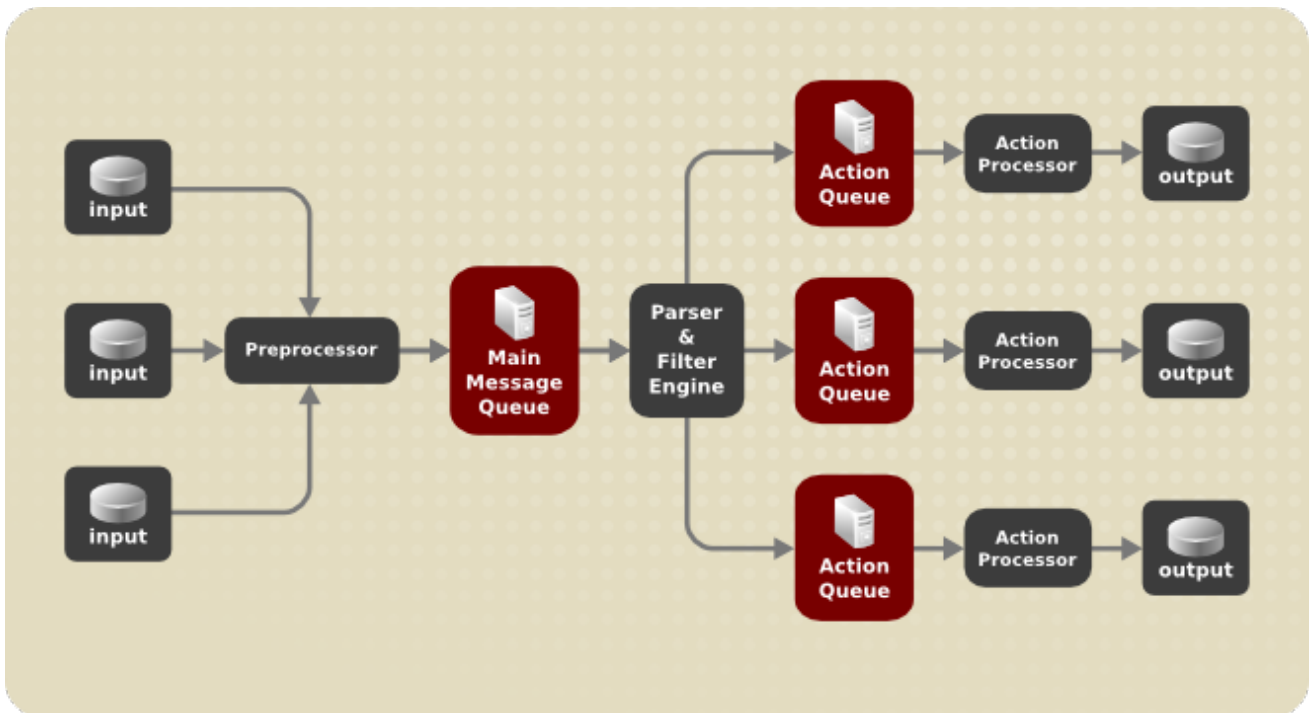
## 25.4.2. Compatibility with sysklogd

The compatibility mode specified via the **-c** option exists in **rsyslog** version 5 but not in version 7. Also, the sysklogd-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 sysklogd-like behavior.

For more information on various **rsyslogd** options, see the **rsyslogd(8)** manual page.

## 25.5. 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 25.1. 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 sections.





## WARNING

If an output plug-in is unable to deliver a message, it is stored in the preceding message queue. If the queue fills, the inputs block until it is no longer full. This will prevent new messages from being logged via the blocked queue. In the absence of separate action queues this can have severe consequences, such as preventing **SSH** logging, which in turn can prevent **SSH** access. Therefore it is advised to use dedicated action queues for outputs which are forwarded over a network or to a database.

### 25.5.1. 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 a 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, with a 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:

```
$SubjectQueueFilename 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 is lost if the computer is power cycled or shut down. However, you can use the **\$ActionQueueSaveOnShutdown** setting to save the data before shutdown. There are two types of in-memory queues:

- *FixedArray* 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, *FixedArray* 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:

```
$SubjectQueueType LinkedList
```

```
$SubjectQueueType 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 combine them in *disk-assisted in-memory queues*. To do so, configure a normal in-memory queue and then add the **\$SubjectQueueFileName** 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*:

```
$SubjectQueueHighWatermark number
```

```
$SubjectQueueLowWatermark 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.

### Example 25.12. Reliable Forwarding of Log Messages to a Server

Rsyslog is often used to maintain a centralized logging system, where log messages are forwarded to a server over the network. To avoid message loss when the server is not available, it is advisable to configure an action queue for the forwarding action. This way, messages that failed to be sent are stored locally until the server is reachable again. Note that such queues are not configurable for connections using the **UDP** protocol. To establish a fully reliable connection, for example when your logging server is outside of your private network, consider using the RELP protocol described in [Section 25.7.4, “Using RELP”](#).

#### Procedure 25.2. Forwarding To a Single Server

Suppose the task is to forward log messages from the system to a server with host name *example.com*, and to configure an action queue to buffer the messages in case of a server outage. To do so, perform the following steps:

- Use the following configuration in **/etc/rsyslog.conf** or create a file with the following content in the **/etc/rsyslog.d/** directory:

```
$ActionQueueType LinkedList
$ActionQueueFileName example_fwd
$ActionResumeRetryCount -1
$ActionQueueSaveOnShutdown on
*. * @@example.com:6514
```

Where:

- **\$ActionQueueType** enables a LinkedList in-memory queue,
- **\$ActionFileName** defines a disk storage, in this case the backup files are created in the **/var/lib/rsyslog/** directory with the *example\_fwd* prefix,
- the **\$ActionResumeRetryCount -1** setting prevents rsyslog from dropping messages when retrying to connect if server is not responding,
- enabled **\$ActionQueueSaveOnShutdown** saves in-memory data if rsyslog shuts down,
- the last line forwards all received messages to the logging server, port specification is optional.

With the above configuration, rsyslog keeps messages in memory if the remote server is not reachable. A file on disk is created only if rsyslog runs out of the configured memory queue space or needs to shut down, which benefits the system performance.

#### Procedure 25.3. Forwarding To Multiple Servers

The process of forwarding log messages to multiple servers is similar to the previous procedure:

- Each destination server requires a separate forwarding rule, action queue specification, and backup file on disk. For example, use the following configuration in `/etc/rsyslog.conf` or create a file with the following content in the `/etc/rsyslog.d/` directory:

```
$ActionQueueType LinkedList
$ActionQueueFileName example_fwd1
$ActionResumeRetryCount -1
$ActionQueueSaveOnShutdown on
*. *          @@example1.com

$ActionQueueType LinkedList
$ActionQueueFileName example_fwd2
$ActionResumeRetryCount -1
$ActionQueueSaveOnShutdown on
*. *          @@example2.com
```

### 25.5.2. Creating a New Directory for rsyslog Log Files

Rsyslog runs as the `syslogd` daemon and is managed by SELinux. Therefore all files to which rsyslog is required to write to, must have the appropriate SELinux file context.

#### Procedure 25.4. Creating a New Working Directory

1. If required to use a different directory to store working files, create a directory as follows:

```
~]# mkdir /rsyslog
```

2. Install utilities to manage SELinux policy:

```
~]# yum install policycoreutils-python
```

3. Set the SELinux directory context type to be the same as the `/var/lib/rsyslog/` directory:

```
~]# semanage fcontext -a -t syslogd_var_lib_t /rsyslog
```

4. Apply the SELinux context:

```
~]# restorecon -R -v /rsyslog
restorecon reset /rsyslog context
unconfined_u:object_r:default_t:s0-
>unconfined_u:object_r:syslogd_var_lib_t:s0
```

5. If required, check the SELinux context as follows:

```
~]# ls -Zd /rsyslog
drwxr-xr-x. root root system_u:object_r:syslogd_var_lib_t:s0
/rsyslog
```

6. Create subdirectories as required. For example:

```
~]# mkdir /rsyslog/work
```

The subdirectories will be created with the same SELinux context as the parent directory.

7. Add the following line in `/etc/rsyslog.conf` immediately before it is required to take effect:

```
$WorkDirectory /rsyslog/work
```

This setting will remain in effect until the next **WorkDirectory** directive is encountered while parsing the configuration files.

### 25.5.3. 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 [the section called “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 a queue on system shutdown.

#### Limiting Queue Size

You can limit the number of messages that queue can contain with the following setting:

```
SubjectQueueHighWatermark 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:

```
SubjectQueueMaxDiscSpace 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*.

```
SubjectQueueDiscardMark 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:

```
SubjectQueueDiscardSeverity 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:

```
$SubjectQueueDequeueTimeBegin hour
```

```
$SubjectQueueDequeueTimeEnd 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:

```
$SubjectQueueWorkerThreadMinimumMessages 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:

```
$SubjectQueueWorkerThreads number
```

where *number* stands for a maximum number of 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:

```
$SubjectQueueWorkerTimeoutThreadShutdown 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:

```
$SubjectQueueDequeueBatchSize number
```

Replace *number* with the maximum number of messages that can be dequeued at once. Note that a higher setting combined with a 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:

```
$SubjectQueueTimeoutShutdown 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:

```
$SubjectQueueTimeoutActionCompletion time
```

In case this timeout expires, any remaining workers are shut down. To save data at shutdown, use:

```
$SubjectQueueTimeoutSaveOnShutdown time
```

If set, all queue elements are saved to disk before **rsyslog** terminates.

#### 25.5.4. Using the New Syntax for rsyslog queues

In the new syntax available in rsyslog 7, queues are defined inside the **action()** object that can be used both separately or inside a ruleset in **/etc/rsyslog.conf**. The format of an action queue is as follows:

```
action(type="action_type" queue.size="queue_size" queue.type="queue_type"
queue.filename="file_name")
```

Replace *action\_type* with the name of the module that is to perform the action and replace *queue\_size* with a maximum number of messages the queue can contain. For *queue\_type*, choose **disk** or select from one of the in-memory queues: **direct**, **linkedlist** or **fixedarray**. For *file\_name* specify only a file name, not a path. Note that if creating a new directory to hold log files, the SELinux context must be set. See [Section 25.5.2, "Creating a New Directory for rsyslog Log Files"](#) for an example.

##### Example 25.13. Defining an Action Queue

To configure the output action with an asynchronous linked-list based action queue which can hold a maximum of 10,000 messages, enter a command as follows:

```
action(type="omfile" queue.size="10000" queue.type="linkedlist"
queue.filename="logfile")
```

The rsyslog 7 syntax for a direct action queues is as follows:

```
*.* action(type="omfile" file="/var/lib/rsyslog/log_file
)
```

The rsyslog 7 syntax for an action queue with multiple parameters can be written as follows:

```
*.* action(type="omfile"
        queue.filename="log_file"
        queue.type="linkedlist"
        queue.size="10000"
)
```

The default work directory, or the last work directory to be set, will be used. If required to use a different work directory, add a line as follows before the action queue:

```
global(workDirectory="/directory")
```

### Example 25.14. Forwarding To a Single Server Using the New Syntax

The following example is based on the procedure [Procedure 25.2, “Forwarding To a Single Server”](#) in order to show the difference between the traditional `sysntax` and the `rsyslog 7` syntax. The `omfwd` plug-in is used to provide forwarding over **UDP** or **TCP**. The default is **UDP**. As the plug-in is built in it does not have to be loaded.

Use the following configuration in `/etc/rsyslog.conf` or create a file with the following content in the `/etc/rsyslog.d/` directory:

```
*.* action(type="omfwd"
        queue.type="linkedlist"
        queue.filename="example_fwd"
        action.resumeRetryCount="-1"
        queue.saveOnShutdown="on"
        target="example.com" port="6514" protocol="tcp"
    )
```

Where:

- `queue.type="linkedlist"` enables a `LinkedList` in-memory queue,
- `queue.filename` defines a disk storage. The backup files are created with the `example_fwd` prefix, in the working directory specified by the preceding `global workDirectory` directive,
- the `action.resumeRetryCount -1` setting prevents `rsyslog` from dropping messages when retrying to connect if server is not responding,
- enabled `queue.saveOnShutdown="on"` saves in-memory data if `rsyslog` shuts down,
- the last line forwards all received messages to the logging server, port specification is optional.

## 25.6. CONFIGURING RSYSLOG ON A LOGGING SERVER

The `rsyslog` service provides facilities both for running a logging server and for configuring individual systems to send their log files to the logging server. See [Example 25.12, “Reliable Forwarding of Log Messages to a Server”](#) for information on client `rsyslog` configuration.

The `rsyslog` service must be installed on the system that you intend to use as a logging server and all systems that will be configured to send logs to it. `Rsyslog` is installed by default in Red Hat Enterprise Linux 6. If required, to ensure that it is, enter the following command as **root**:

```
~]# yum install rsyslog
```



The default protocol and port for syslog traffic is **UDP** and **514**, as listed in the `/etc/services` file. However, **rsyslog** defaults to using **TCP** on port **514**. In the configuration file, `/etc/rsyslog.conf`, **TCP** is indicated by `@@`.

Other ports are sometimes used in examples, however SELinux is only configured to allow sending and receiving on the following ports by default:

```
~]# semanage port -l | grep syslog
syslogd_port_t          tcp          6514, 601
syslogd_port_t          udp          514, 6514, 601
```

The **semanage** utility is provided as part of the `policycoreutils-python` package. If required, install the package as follows:

```
~]# yum install policycoreutils-python
```

In addition, by default the SELinux type for **rsyslog**, `rsyslogd_t`, is configured to permit sending and receiving to the remote shell (**rsh**) port with SELinux type `rsh_port_t`, which defaults to **TCP** on port **514**. Therefore it is not necessary to use **semanage** to explicitly permit **TCP** on port **514**. For example, to check what SELinux is set to permit on port **514**, enter a command as follows:

```
~]# semanage port -l | grep 514
output omitted
rsh_port_t             tcp          514
syslogd_port_t         tcp          6514, 601
syslogd_port_t         udp          514, 6514, 601
```

For more information on SELinux, see [Red Hat Enterprise Linux 6 SELinux User Guide](#).

Perform the steps in the following procedures on the system that you intend to use as your logging server. All steps in these procedure must be made as the **root** user.

### Procedure 25.5. Configure SELinux to Permit rsyslog Traffic on a Port

If required to use a new port for **rsyslog** traffic, follow this procedure on the logging server and the clients. For example, to send and receive **TCP** traffic on port **10514**, proceed as follows:

1. 

```
~]# semanage port -a -t syslogd_port_t -p tcp 10514
```

2. Review the SELinux ports by entering the following command:

```
~]# semanage port -l | grep syslog
```

3. If the new port was already configured in `/etc/rsyslog.conf`, restart **rsyslog** now for the change to take effect:

```
~]# service rsyslog restart
```

4. Verify which ports **rsyslog** is now listening to:

```
~]# netstat -tnlp | grep rsyslog
tcp        0          0 0.0.0.0:10514        0.0.0.0:*    LISTEN
2528/rsyslogd
```

```
tcp          0          0 :::10514          :::*          LISTEN
2528/rsyslog
```

See the **semanage-port(8)** manual page for more information on the **semanage port** command.

### Procedure 25.6. Configuring The iptables Firewall

Configure the **iptables** firewall to allow incoming **rsyslog** traffic. For example, to allow **TCP** traffic on port **10514**, proceed as follows:

1. Open the **/etc/sysconfig/iptables** file in a text editor.
2. Add an **INPUT** rule allowing **TCP** traffic on port **10514** to the file. The new rule must appear before any **INPUT** rules that **REJECT** traffic.

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 10514 -j ACCEPT
```

3. Save the changes to the **/etc/sysconfig/iptables** file.
4. Restart the **iptables** service for the firewall changes to take effect.

```
~]# service iptables restart
```

### Procedure 25.7. Configuring rsyslog to Receive and Sort Remote Log Messages

1. Open the **/etc/rsyslog.conf** file in a text editor and proceed as follows:
  - a. Add these lines below the modules section but above the **Provides UDP syslog reception** section:

```
# Define templates before the rules that use them

### Per-Host Templates for Remote Systems ###
$template TmplAuthpriv,
"/var/log/remote/auth/%HOSTNAME%/%PROGRAMNAME:::secpath-
replace%.log"
$template TmplMsg,
"/var/log/remote/msg/%HOSTNAME%/%PROGRAMNAME:::secpath-
replace%.log"
```

- b. Replace the default **Provides TCP syslog reception** section with the following:

```
# Provides TCP syslog reception
$ModLoad imtcp
# Adding this ruleset to process remote messages
$RuleSet remote1
authpriv.* ?TmplAuthpriv
*.info;mail.none;authpriv.none;cron.none ?TmplMsg
$RuleSet RSYSLOG_DefaultRuleset #End the rule set by switching
back to the default rule set
$InputTCPServerBindRuleset remote1 #Define a new input and bind
it to the "remote1" rule set
$InputTCPServerRun 10514
```

-

Save the changes to the `/etc/rsyslog.conf` file.

2. The **rsyslog** service must be running on both the logging server and the systems attempting to log to it.
  - a. Use the **service** command to start the **rsyslog** service.

```
~]# service rsyslog start
```

- b. To ensure the **rsyslog** service starts automatically in future, enter the following command as root:

```
~]# chkconfig rsyslog on
```

Your log server is now configured to receive and store log files from the other systems in your environment.

### 25.6.1. Using The New Template Syntax on a Logging Server

Rsyslog 7 has a number of different templates styles. The string template most closely resembles the legacy format. Reproducing the templates from the example above using the string format would look as follows:

```
template(name="TplAuthpriv" type="string"
          string="/var/log/remote/auth/%HOSTNAME%/%PROGRAMNAME:::secpath-
replace%.log"
          )

template(name="TplMsg" type="string"
          string="/var/log/remote/msg/%HOSTNAME%/%PROGRAMNAME:::secpath-
replace%.log"
          )
```

These templates can also be written in the list format as follows:

```
template(name="TplAuthpriv" type="list") {
    constant(value="/var/log/remote/auth/")
    property(name="hostname")
    constant(value="/")
    property(name="programname" SecurePath="replace")
    constant(value=".log")
}

template(name="TplMsg" type="list") {
    constant(value="/var/log/remote/msg/")
    property(name="hostname")
    constant(value="/")
    property(name="programname" SecurePath="replace")
    constant(value=".log")
}
```

This template text format might be easier to read for those new to rsyslog and therefore can be easier to adapt as requirements change.

To complete the change to the new syntax, we need to reproduce the module load command, add a rule set, and then bind the rule set to the protocol, port, and ruleset:

```
module(load="imtcp")

ruleset(name="remote1"){
    authpriv.*    action(type="omfile" DynaFile="TplAuthpriv")
    *.info;mail.none;authpriv.none;cron.none action(type="omfile"
DynaFile="TplMsg")
}

input(type="imtcp" port="10514" ruleset="remote1")
```

## 25.7. 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**.
- **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**, and so on.
- **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 25.3.3, “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](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.

## 25.7.1. 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 imported messages by replacing *severity* with an appropriate keyword. Replace *facility* with a keyword to define the subsystem that produced the message. The keywords for severity and facility are the same as those used in facility/priority-based filters, see [Section 25.3.1, “Filters”](#).

### Example 25.15. 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:
$InputStateFile state-apache-error
$InputRunFileMonitor
```

## 25.7.2. 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, choose 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 25.16. 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.

## 25.7.3. 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.

#### 25.7.4. Using RELP

*Reliable Event Logging Protocol* (RELP) is a networking protocol for data logging in computer networks. It is designed to provide reliable delivery of event messages, which makes it useful in environments where message loss is not acceptable.

To configure RELP, first install the `rsyslog-relp` package both on the server and the client:

```
~]# yum install rsyslog-relp
```

Then, configure both the server and the client.

1. To configure the client, configure:

- loading the required modules
- the TCP input port
- the transport settings

by adding the following configuration to the `/etc/rsyslog.conf` file:

```
$ModLoad omrelp
$ModLoad imuxsock
$ModLoad imtcp
$InputTCPServerRun "port"
*. * :omrelp:"target_IP":"target_port"
```

Replace *port* to start a listener at the required port.

Replace *target\_IP* and *target\_port* with the IP address and port that identify the target server.

2. To configure the server:

- configure loading the modules
- configure the TCP input similarly to the client configuration
- configure the rules and choose an action to be performed

by adding the following configuration to the `/etc/rsyslog.conf` file:

```
$ModLoad imuxsock
$ModLoad imrelp
$RuleSet relp
*. * "log_path"
$InputRELPServerBindRuleset relp
$InputRELPServerRun "target_port"
```

Replace *target\_port* with the same value as on the clients.

In the previous example, *log\_path* specifies the path for storing messages.

## 25.8. 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.

## 25.9. MANAGING LOG FILES IN A GRAPHICAL ENVIRONMENT

As an alternative to the aforementioned command-line utilities, Red Hat Enterprise Linux 6 provides an accessible GUI for managing log messages.

### 25.9.1. 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**.

#### NOTE

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
```

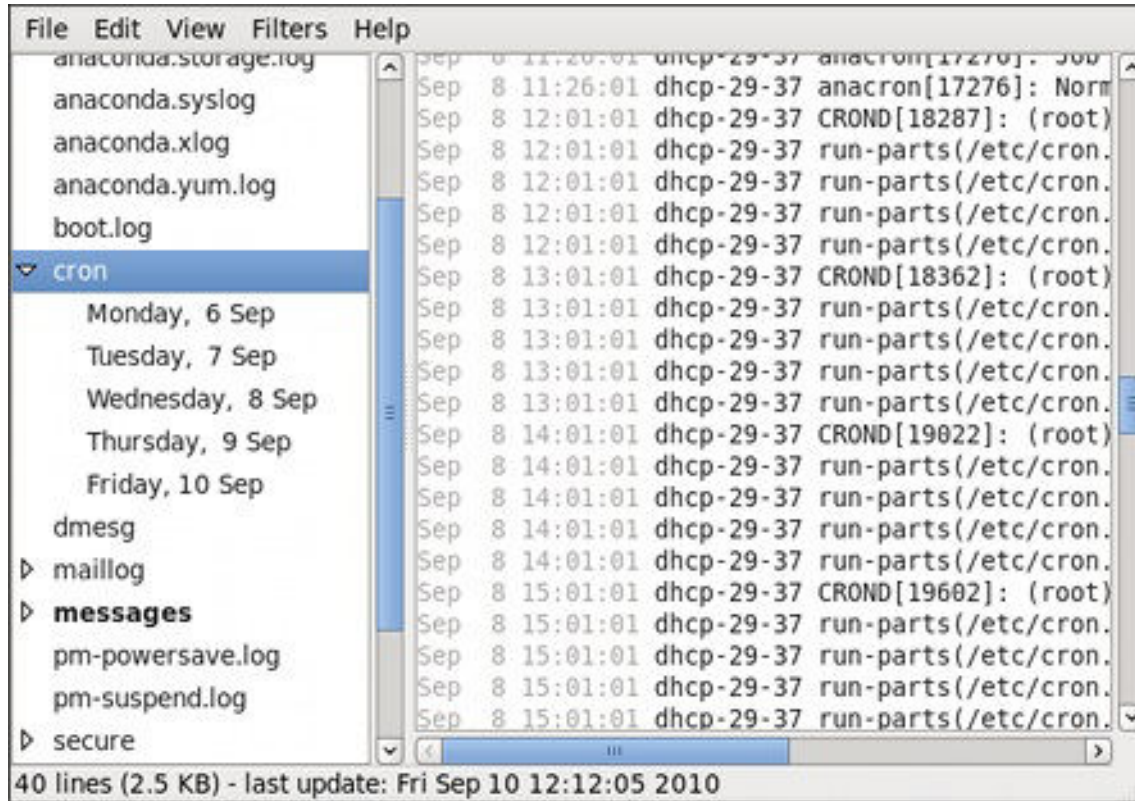
The `gnome-system-log` package is provided by the Optional subscription channel that must be enabled before installation. See [Section 8.4.8, "Adding the Optional and Supplementary Repositories"](#) for more information on Red Hat additional channels. For more information on installing packages with Yum, see [Section 8.2.4, "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:

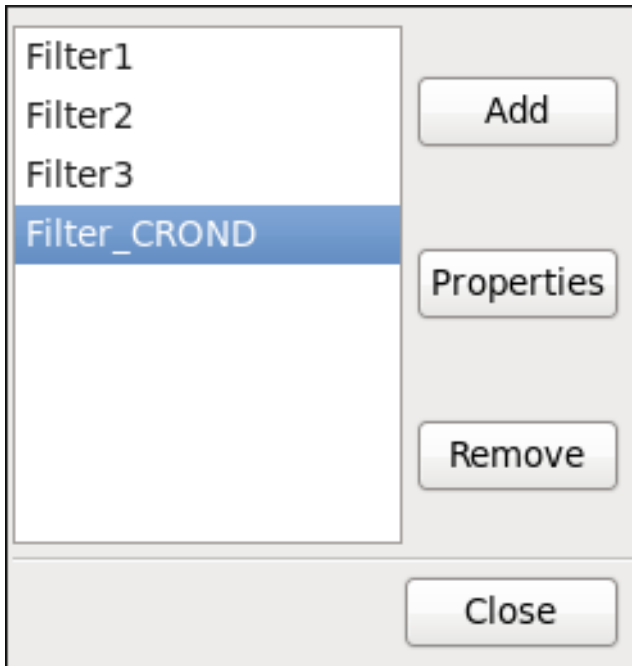
```
~]$ gnome-system-log
```

The application only displays log files that exist; thus, the list might differ from the one shown in [Figure 25.2, “Log File Viewer”](#).



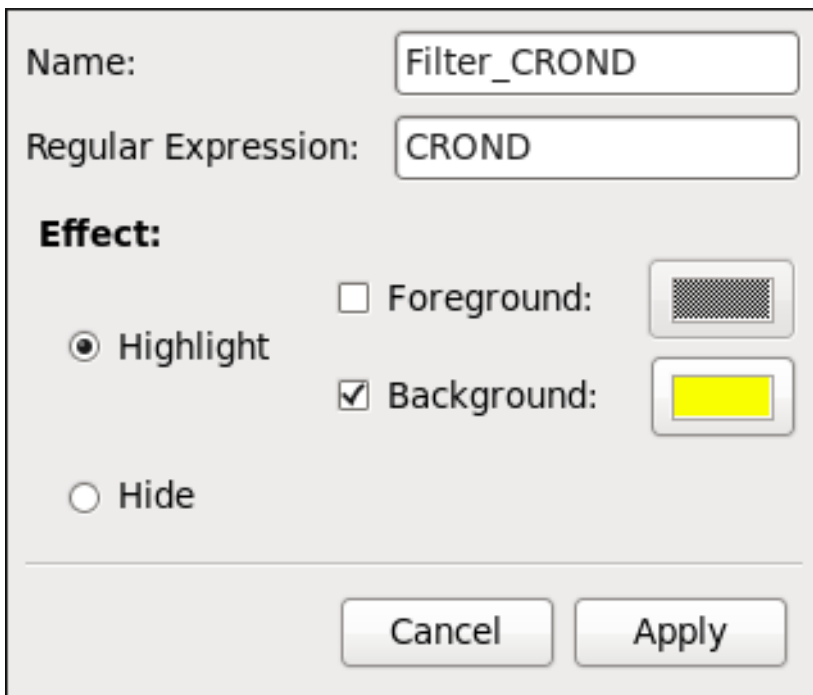
**Figure 25.2. Log File Viewer**

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 25.3. Log File Viewer - Filters**

Adding or editing a filter lets you define its parameters as is shown in [Figure 25.4, “Log File Viewer - defining a filter”](#).



**Figure 25.4. 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.

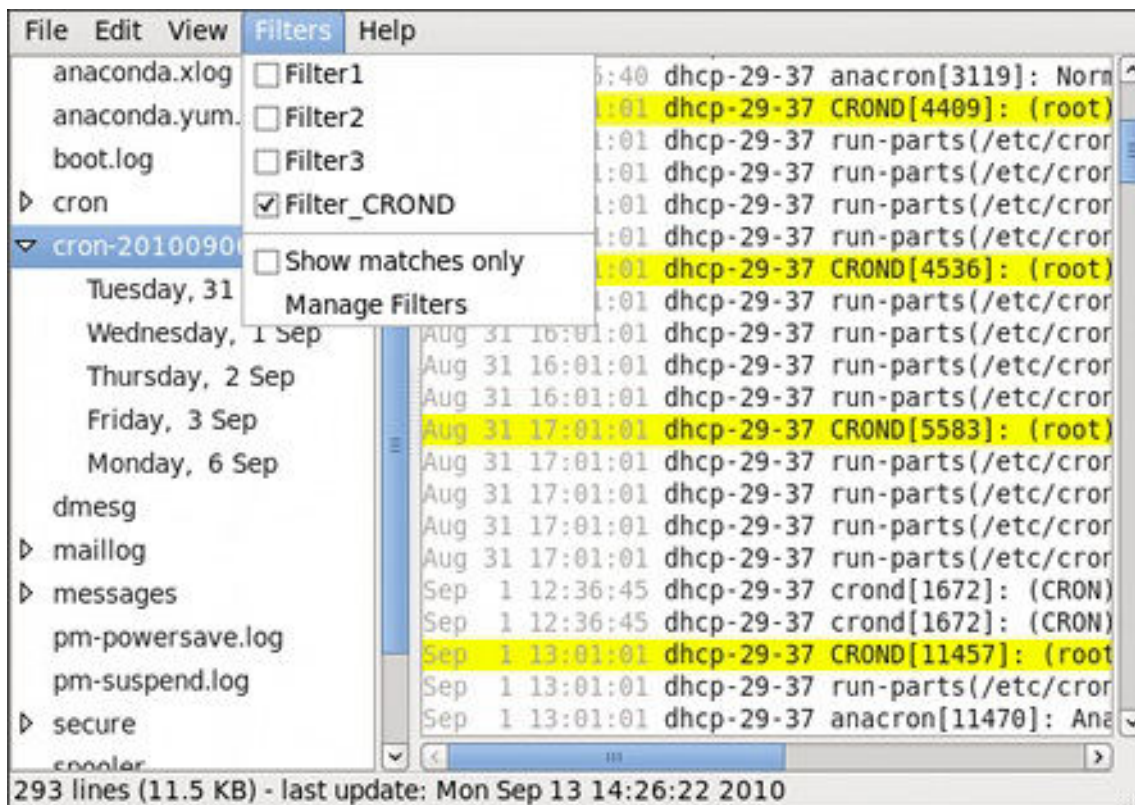
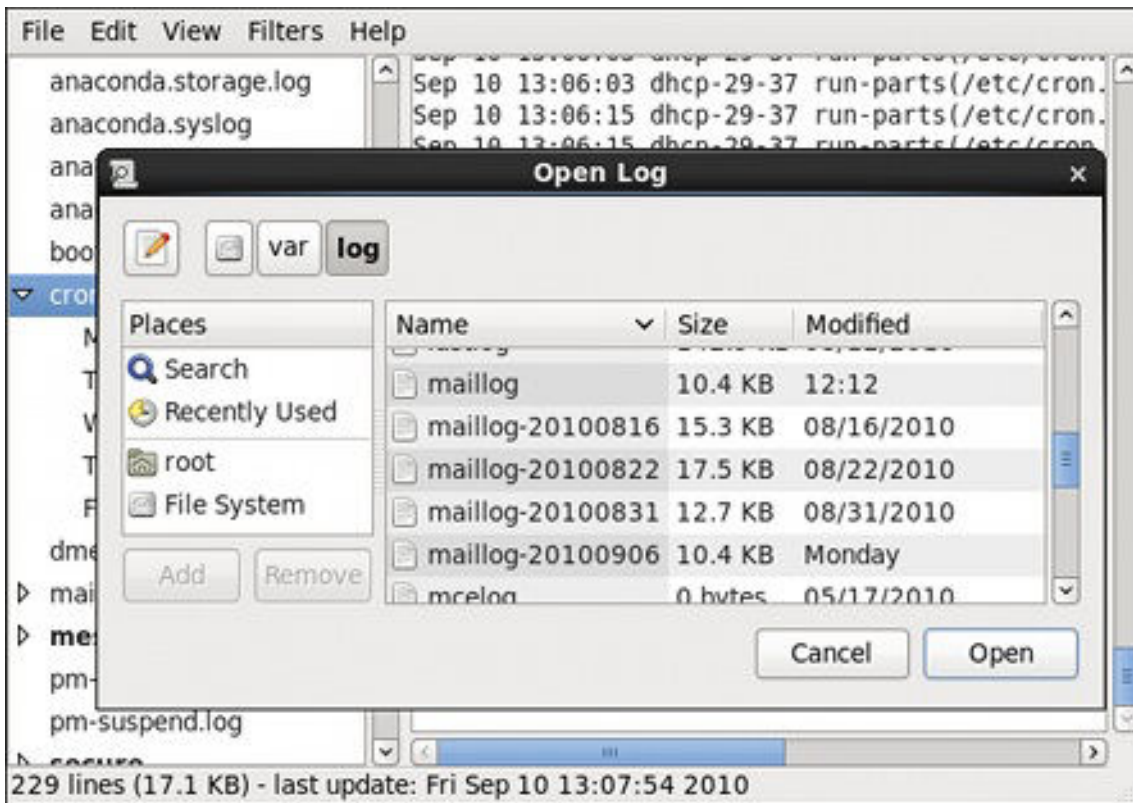


Figure 25.5. Log File Viewer - enabling a filter

When you select the **Show matches only** option, only the matched strings will be shown in the log file you are currently viewing.

### 25.9.2. Adding a Log File

To add a log file you want 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 want to view. [Figure 25.6, “Log File Viewer - adding a log file”](#) illustrates the **Open Log** window.



**Figure 25.6. 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.



#### NOTE

The **Log File Viewer** also allows you to open log files zipped in the **.gz** format.

### 25.9.3. 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 25.7, “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.



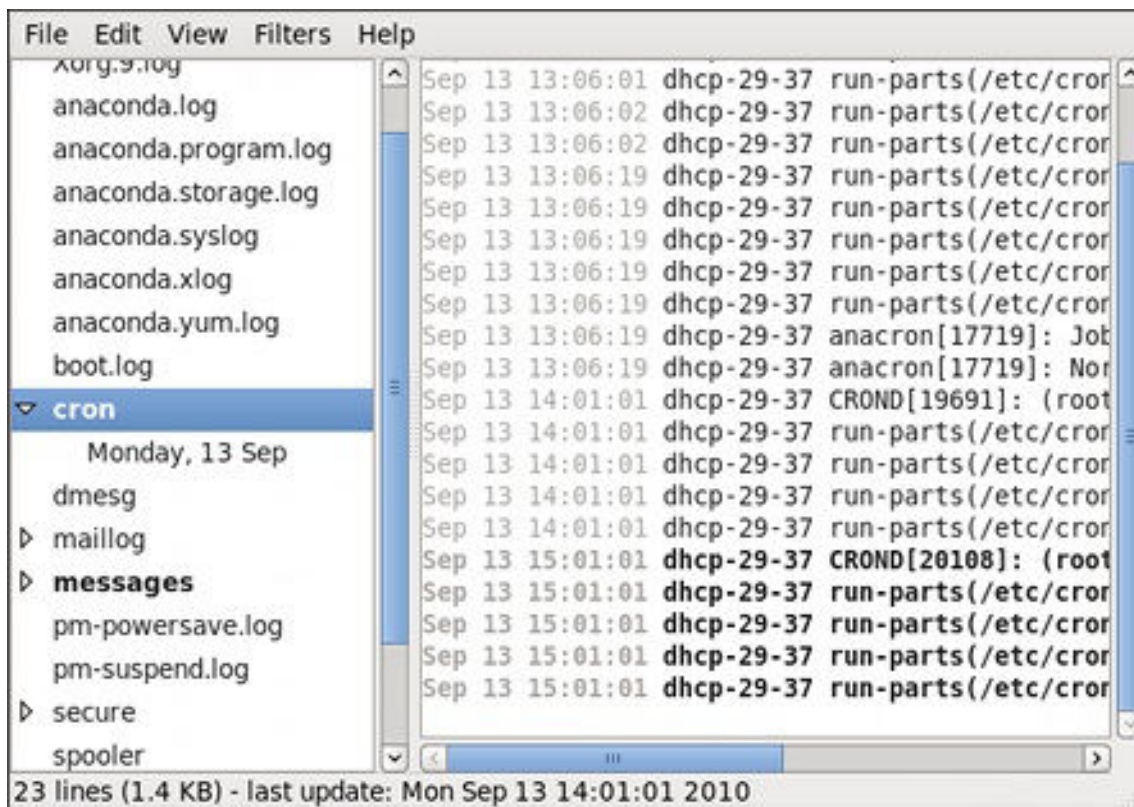


Figure 25.7. Log File Viewer - new log alert

## 25.10. 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.

### Online Documentation

The **rsyslog** home page offers additional documentation, configuration examples, and video tutorials. Make sure to consult the documents relevant to the version you are using:

- [rsyslog version 5 documentation on the rsyslog home page](#) — The default version of **rsyslog** in Red Hat Enterprise Linux 6 is version 5.
- [rsyslog version 7 documentation on the rsyslog home page](#) — Version 7 of **rsyslog** is available for Red Hat Enterprise Linux 6 in the **rsyslog7** package.
- [Description of queues on the rsyslog Home Page](#) — General information on various types of message queues and their usage.

### See Also

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

and **sudo** commands.

## CHAPTER 26. UPGRADING MYSQL

Red Hat is committed to fully supporting the upstream version of MySQL, which is currently included in Red Hat Enterprise Linux, until the end of production phase 3, as long as upstream security and bug fixes are available. For overview of Red Hat Enterprise Linux Life Cycle, see [https://access.redhat.com/support/policy/updates/errata#Production\\_2\\_Phase](https://access.redhat.com/support/policy/updates/errata#Production_2_Phase).

More recent versions of MySQL, MySQL 5.6 and MySQL 5.7, are provided as the rh-mysql56 and rh-mysql57 Software Collections. These components are part of Red Hat Software Collections, available for all supported releases of Red Hat Enterprise Linux 6 on AMD64 and Intel 64 architectures.

For information on how to get access to Red Hat Software Collections, see the [Red Hat Software Collections Release Notes](#).

See the [Red Hat Software Collections Product Life Cycle](#) document for information regarding length of support for individual components.

Note that you cannot directly migrate from MySQL 5.1 to the currently supported versions. Refer to detailed procedures how to migrate [from MySQL 5.1 to MySQL 5.5](#), [from MySQL 5.5 to MySQL 5.6](#), and [from MySQL 5.6 to MySQL 5.7](#).

## CHAPTER 27. 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 (see [Section 27.1, “Cron and Anacron”](#) and [Section 27.2, “At and Batch”](#) respectively).

### 27.1. 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.

#### 27.1.1. Installing Cron and Anacron

To install Cron and Anacron, you need to install the `crontab` package with Cron and the `crontab-anacron` package with Anacron (`crontab-anacron` is a sub-package of `crontab`).

To determine if the packages are already installed on your system, issue the `rpm -q crontab crontab-anacron` command. The command returns full names of the `crontab` and `crontab-anacron` packages if already installed or notifies you that the packages are not available.

To install the packages, use the `yum` command in the following form:

```
yum install package
```

For example, to install both Cron and Anacron, type the following at a shell prompt:

```
~]# yum install crontab crontab-anacron
```

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 8.2.4, “Installing Packages”](#).

#### 27.1.2. 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 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 12, Services and Daemons](#).

### 27.1.2.1. Starting and Stopping the Cron Service

To determine if the service is running, use the command `service crond status`.

To run the `crond` service in the current session, type the following at a shell prompt as `root`:

```
service crond start
```

To configure the service to be automatically started at boot time, use the following command:

```
chkconfig crond on
```

This command enables the service in runlevel 2, 3, 4, and 5. Alternatively, you can use the **Service Configuration** utility as described in [Section 12.2.1.1, “Enabling and Disabling a Service”](#).

### 27.1.2.2. Stopping the Cron Service

To stop the `crond` service, type the following at a shell prompt as `root`:

```
service crond stop
```

To disable starting the service at boot time, use the following command:

```
chkconfig crond off
```

This command disables the service in all runlevels. Alternatively, you can use the **Service Configuration** utility as described in [Section 12.2.1.1, “Enabling and Disabling a Service”](#).

### 27.1.2.3. Restarting the Cron Service

To restart the `crond` service, type the following at a shell prompt:

```
service crond restart
```

This command stops the service and starts it again in quick succession.

## 27.1.3. 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** — user name 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.

### 27.1.3.1. 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).

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.

### 27.1.4. Configuring Cron Jobs

The configuration file for cron jobs is the `/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 27.1.3, “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, 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 `crontab` 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 user, login as that user and type the command `crontab -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 user name and written to the file `/var/spool/cron/username`. To list the contents of your crontab file, use the `crontab -l` command.

**NOTE**

Do not specify the user when defining a job with the **crontab** utility.

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.

**NOTE**

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.

### 27.1.5. 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**).

### 27.1.6. 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 **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 27.1.5, “Controlling Access to Cron”](#).

## 27.2. 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.

### 27.2.1. Installing At and Batch

To determine if the `at` package is already installed on your system, issue the `rpm -q at` command. 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:

```
yum install package
```

To install At and Batch, type the following at a shell prompt:

```
~]# yum install at
```

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 8.2.4, “Installing Packages”](#).

### 27.2.2. 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 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 12, Services and Daemons](#).

#### 27.2.2.1. Starting and Stopping the At Service

To determine if the service is running, use the command `service atd status`.

To run the `atd` service in the current session, type the following at a shell prompt as **root**:

```
service atd start
```

To configure the service to start automatically at boot, use the following command:

```
chkconfig atd on
```



#### NOTE

It is recommended to start the service at boot automatically.

This command enables the service in runlevel 2, 3, 4, and 5. Alternatively, you can use the **Service Configuration** utility as described in [Section 12.2.1.1, “Enabling and Disabling a Service”](#).

### 27.2.2.2. Stopping the At Service

To stop the **atd** service, type the following at a shell prompt as **root**

```
service atd stop
```

To disable starting the service at boot time, use the following command:

```
chkconfig atd off
```

This command disables the service in all runlevels. Alternatively, you can use the **Service Configuration** utility as described in [Section 12.2.1.1, “Enabling and Disabling a Service”](#).

### 27.2.2.3. Restarting the At Service

To restart the **atd** service, type the following at a shell prompt:

```
service atd restart
```

This command stops the service and starts it again in quick succession.

## 27.2.3. 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:

- o **HH:MM** specifies the exact hour and minute; For example, **04:00** specifies 4:00 a.m.
- o **midnight** specifies 12:00 a.m.
- o **noon** specifies 12:00 p.m.
- o **teatime** specifies 4:00 p.m.
- o **MONTHDAYYEAR** format; For example, **January 15 2012** specifies the 15th day of January in the year 2012. The year value is optional.
- o **MMDDYY**, **MM/DD/YY**, or **MM.DD.YY** formats; For example, **011512** for the 15th day of January in the year 2012.
- o **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 27.2.5, “Viewing Pending Jobs”](#) for more information.

You can also restrict the usage of the **at** command. For more information, see [Section 27.2.7, “Controlling Access to At and Batch”](#) for details.

### 27.2.4. 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:
  - 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.

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 27.2.5, “Viewing Pending Jobs”](#) for more information.

You can also restrict the usage of the **batch** command. For more information, see [Section 27.2.7, “Controlling Access to At and Batch”](#) for details.

### 27.2.5. 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.

### 27.2.6. Additional Command-Line Options

Additional command-line options for **at** and **batch** include the following:

**Table 27.1. at and batch Command-Line Options**

Option	Description
<b>-f</b>	Read the commands or shell script from a file instead of specifying them at the prompt.
<b>-m</b>	Send email to the user when the job has been completed.
<b>-v</b>	Display the time that the job is executed.

### 27.2.7. 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.

## 27.3. ADDITIONAL RESOURCES

To learn more about configuring automated tasks, see the following installed documentation:

- **cron** 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** manual page contains an overview of anacron.
- **anacrontab** manual page contains an overview of the **anacrontab** file.

- `/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 28. AUTOMATIC BUG REPORTING TOOL (ABRT)

The **Automatic Bug Reporting Tool**, commonly abbreviated as **ABRT**, consists of the **abrt** daemon and a number of system services and utilities to process, analyze, and report 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. For a brief overview of the most important ABRT components, see [Table 28.1, “Basic ABRT components”](#).



### IMPORTANT

For Red Hat Enterprise Linux 6.2, the Automatic Bug Reporting Tool has been upgraded to version 2.0. The **ABRT** 2-series brings major improvements to automatic bug detection and reporting.

**Table 28.1. Basic ABRT components**

Component	Package	Description
<b>abrt</b>	abrt	The <b>ABRT</b> daemon which runs under the root user as a background service.
<b>abrt-applet</b>	abrt-gui	The program that receives messages from <b>abrt</b> and informs you whenever a new problem occurs.
<b>abrt-gui</b>	abrt-gui	The GUI application that shows collected problem data and allows you to further process it.
<b>abrt-cli</b>	abrt-cli	The command-line interface that provides similar functionality to the GUI.
<b>abrt-ccpp</b>	abrt-addon-ccpp	The <b>ABRT</b> service that provides the C/C++ problems analyzer.
<b>abrt-oops</b>	abrt-addon-kerneloops	The <b>ABRT</b> service that provides the kernel oopses analyzer.
<b>abrt-vmcore</b>	abrt-addon-vmcore <sup>[a]</sup>	The <b>ABRT</b> service that provides the kernel panic analyzer and reporter.

[a] The abrt-addon-vmcore package is provided by the Optional subscription channel. See [Section 8.4.8, “Adding the Optional and Supplementary Repositories”](#) for more information on Red Hat additional channels.

**ABRT** currently supports detection of crashes in applications written in the C/C++ and Python languages, as well as kernel oopses. With Red Hat Enterprise Linux 6.3, **ABRT** can also detect kernel panics if the additional abrt-addon-vmcore package is installed and the **kdump** crash dumping mechanism is enabled and configured on the system accordingly.

**ABRT** is capable of reporting problems to a remote issue tracker. Reporting can be configured to

happen automatically whenever an issue is detected, or problem data can be stored locally, reviewed, reported, and deleted manually by a user. The reporting tools can send problem data to a Bugzilla database, a Red Hat Technical Support (RHTSupport) site, upload it using **FTP/SCP**, email it, or write it to a file.

The part of **ABRT** which handles already-existing problem data (as opposed to, for example, creation of new problem data) has been factored out into 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**. However, **ABRT** and **libreport** operation and configuration is closely integrated. They are therefore discussed as one in this document.

Whenever a problem is detected, **ABRT** compares it with all existing problem data and determines whether that same problem has been recorded. If it has been, the existing problem data is updated and the most recent (duplicate) problem is not recorded again. If this 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 analysis 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 the crashed kernel.

After the problem directory is created and problem data gathered, you can further process, analyze and report the problem using either the **ABRT** GUI, or the **abrt-cli** utility for the command line. For more information about these tools, see [Section 28.2, “Using the Graphical User Interface”](#) and [Section 28.3, “Using the Command-Line Interface”](#) respectively.

## NOTE

If you do not use **ABRT** to further analyze and report the detected problems but instead you report problems using a legacy problem reporting tool, **report**, note that you can no longer file new bugs. The **report** utility can now only be used to attach new content to the already existing bugs in the RHTSupport or Bugzilla database. Use the following command to do so:

```
report [-v] --target target --ticket ID file
```

...where *target* is either **strata** for reporting to RHTSupport or **bugzilla** for reporting to Bugzilla. *ID* stands for number identifying an existing problem case in the respective database, and *file* is a file containing information to be added to the problem case.

If you want to report new problems and you do not want to use **abrt-cli**, you can now use the **report-cli** utility instead of **report**. Issue the following command to let **report-cli** to guide you through the problem reporting process:

```
report-cli -r dump_directory
```

...where *dump\_directory* is a problem data directory created by **ABRT** or some other application using **libreport**. For more information on **report-cli**, see **man report-cli**.

## 28.1. INSTALLING ABRT AND STARTING ITS SERVICES

As a prerequisite for its use, the **abrt** daemon requires the **abrt** user to exist for file system

operations in the `/var/spool/abrt` directory. When the `abrt` package is installed, it automatically creates the `abrt` user whose UID and GID is 173, if such user does not already exist. Otherwise, the `abrt` user can be created manually. In that case, any UID and GID can be chosen, because `abrt` does not require a specific UID and GID.

As the first step in order to use **ABRT**, you should ensure that the `abrt-desktop` package is installed on your system by running the following command as the root user:

```
~]# yum install abrt-desktop
```

With `abrt-desktop` installed, you will be able to use **ABRT** only in its graphical interface. If you intend to use **ABRT** on the command line, install the `abrt-cli` package:

```
~]# yum install abrt-cli
```

See [Section 8.2.4, “Installing Packages”](#) for more information on how to install packages with the **Yum** package manager.

Your next step should be to verify that `abrt` is running. The daemon is typically configured to start up at boot time. You can use the following command as root to verify its current status:

```
~]# service abrt status
abrt (pid 1535) is running...
```

If the `service` command returns the `abrt is stopped` message, the daemon is not running. It can be started for the current session by entering this command:

```
~]# service abrt start
Starting abrt daemon: [ OK ]
```

Similarly, you can follow the same steps to check and start up the `abrt-ccpp` service if you want **ABRT** to catch C/C++ crashes. To set **ABRT** to detect kernel oopses, use the same steps for the `abrt-oops` service. Note that this service cannot catch kernel oopses which cause the system to fail, to become unresponsive or to reboot immediately. To be able to detect such kernel oopses with **ABRT**, you need to install the `abrt-vmcore` service. If you require this functionality, see [Section 28.4.5, “Configuring ABRT to Detect a Kernel Panic”](#) for more information.

When installing **ABRT** packages, all respective ABRT services are automatically enabled for **runlevels 3 and 5**. You can disable or enable any ABRT service for the desired runlevels using the `chkconfig` utility. See [Section 12.2.3, “Using the chkconfig Utility”](#) for more information.



### WARNING

Please note that installing **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
```

Finally, if you run ABRT in a graphical desktop environment, you can verify that the **ABRT notification applet** is running:

```
~]$ ps -e1 | grep abrt-applet
0 S    500   2036  1824   0  80    0 - 61604 poll_s ?      00:00:00 abrt-
applet
```

If the **ABRT** notification applet is not running, you can start it manually in your current desktop session by running the **abrt-applet** program:

```
~]$ abrt-applet &
[1] 2261
```

The applet can be configured to start automatically when your graphical desktop session starts. You can ensure that the **ABRT** notification applet is added to the list of programs and selected to run at system startup by selecting the **System** → **Preferences** → **Startup Applications** menu in the top panel.

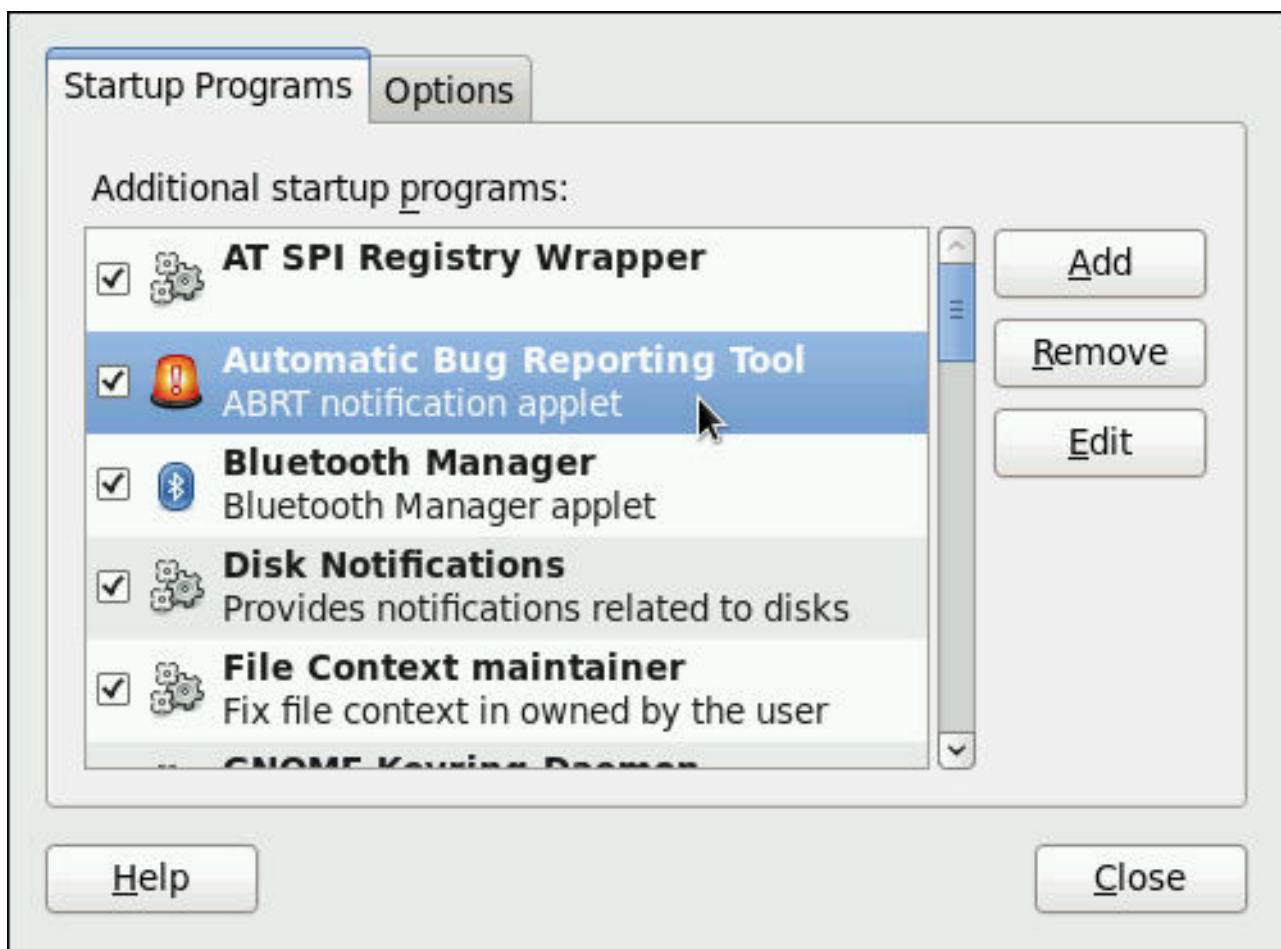


Figure 28.1. Setting ABRT notification applet to run automatically.

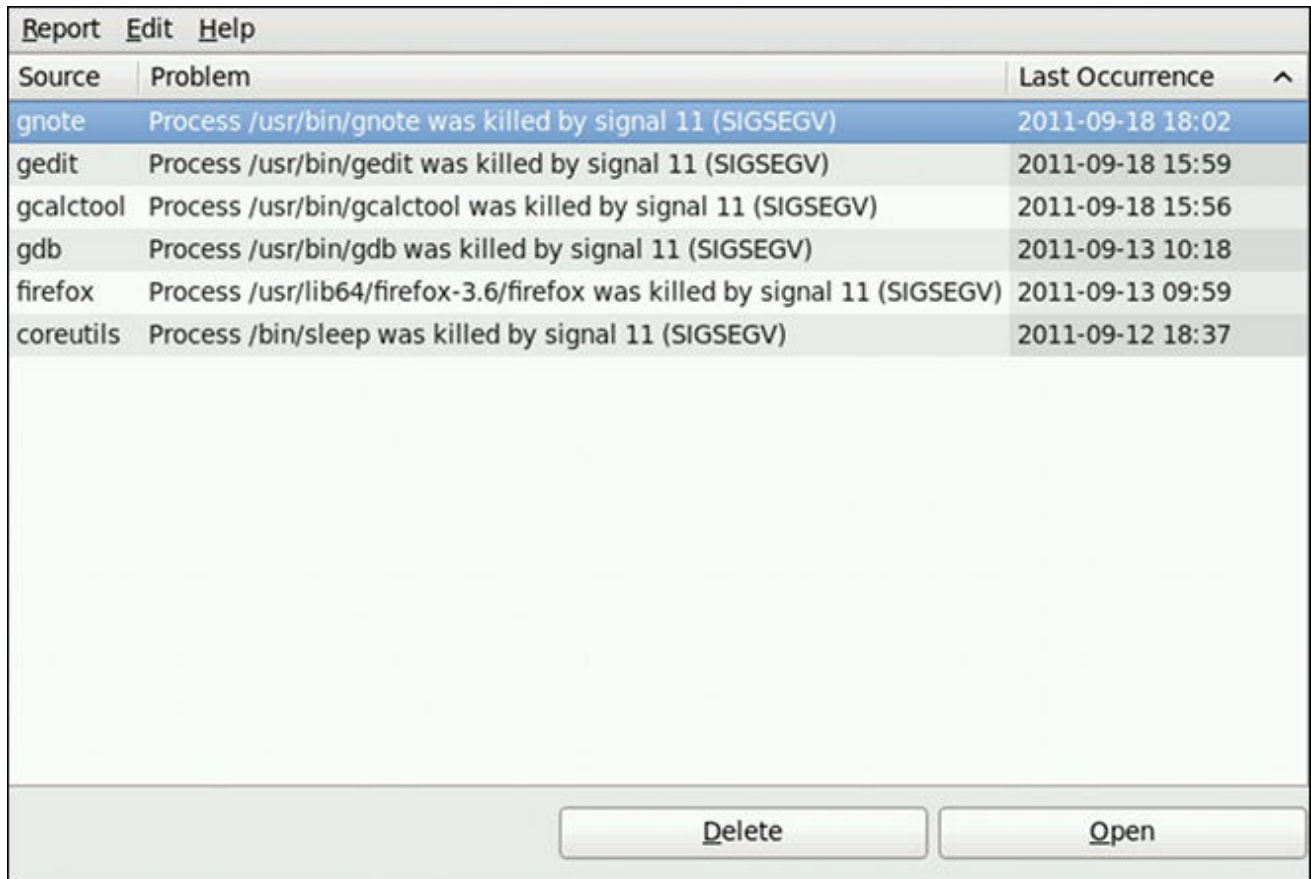
## 28.2. USING THE GRAPHICAL USER INTERFACE

The **ABRT** daemon sends a broadcast D-Bus message whenever a problem report is created. If the **ABRT** notification applet is running, it catches this message and displays an orange alarm icon in the Notification Area. You can open the **ABRT GUI** application using this icon. As an alternative, you can display the **ABRT** GUI by selecting the **Application** → **System Tools** → **Automatic Bug Reporting Tool** menu item.

Alternatively, you can run the **ABRT** GUI from the command line as follows:

```
~]$ abrt-gui &
```

The **ABRT** GUI provides an easy and intuitive way of viewing, reporting and deleting of reported problems. The **ABRT** 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.

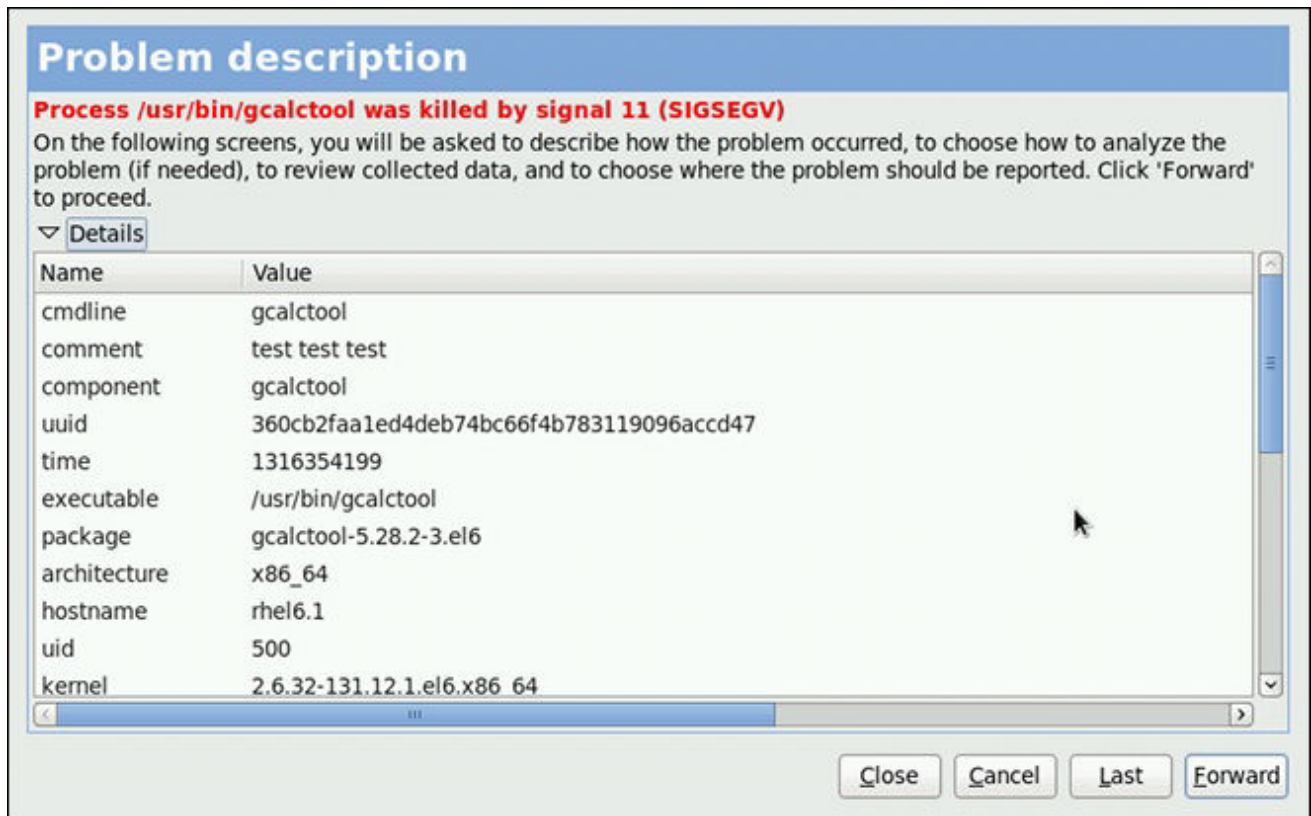


The screenshot shows a window titled 'Report Edit Help'. It contains a table with three columns: 'Source', 'Problem', and 'Last Occurrence'. The table lists several reported problems, each with a source application name, a description of the crash, and the date and time of the last occurrence. At the bottom of the window, there are two buttons: 'Delete' and 'Open'.

Source	Problem	Last Occurrence
gnote	Process /usr/bin/gnote was killed by signal 11 (SIGSEGV)	2011-09-18 18:02
gedit	Process /usr/bin/gedit was killed by signal 11 (SIGSEGV)	2011-09-18 15:59
gcalctool	Process /usr/bin/gcalctool was killed by signal 11 (SIGSEGV)	2011-09-18 15:56
gdb	Process /usr/bin/gdb was killed by signal 11 (SIGSEGV)	2011-09-13 10:18
firefox	Process /usr/lib64/firefox-3.6/firefox was killed by signal 11 (SIGSEGV)	2011-09-13 09:59
coreutils	Process /bin/sleep was killed by signal 11 (SIGSEGV)	2011-09-12 18:37

**Figure 28.2.** An example of running **ABRT** GUI.

If you double-click on a problem report line, you can access the detailed problem description and proceed with the process of determining how the problem should be analyzed, and where it should be reported.



**Figure 28.3. A detailed problem data example**

You are first asked to provide additional information about the problem which occurred. You should provide detailed information on how the problem happened and what steps should be done in order to reproduce it. In the next steps, choose how the problem will be analyzed and generate a backtrace depending on your configuration. You can skip the analysis and backtrace-generation steps but remember that developers need as much information about the problem as possible. You can always modify the backtrace and remove any sensitive information you do not want to provide before you send the problem data out.



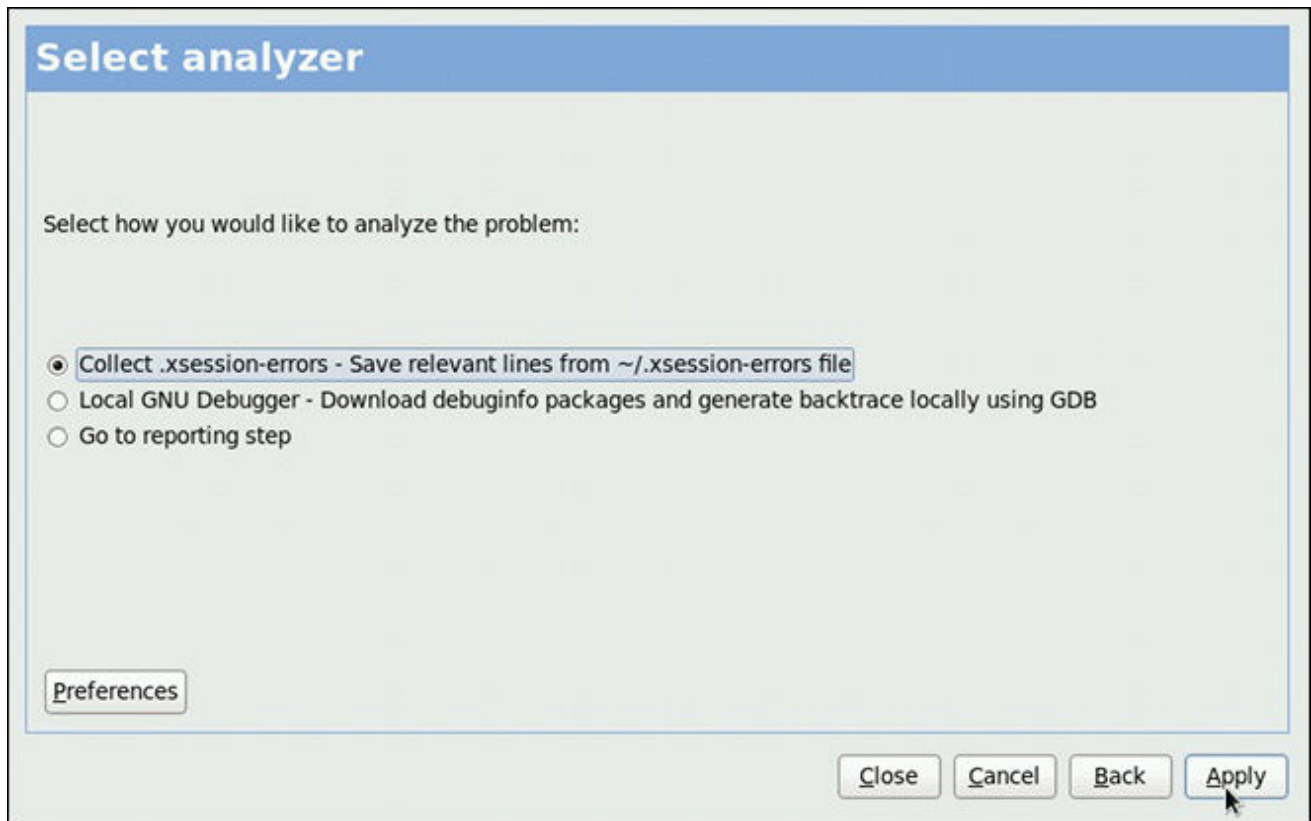


Figure 28.4. Selecting how to analyze the problem

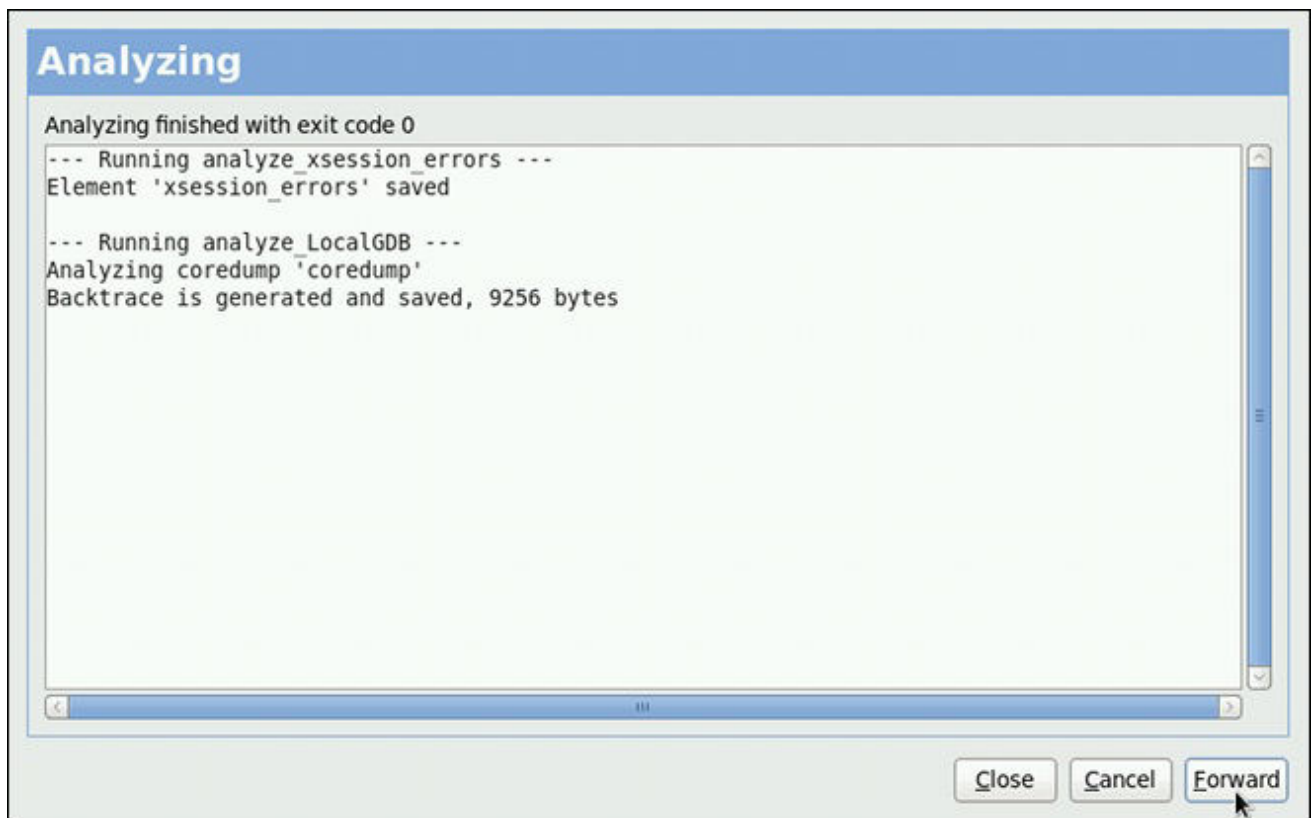


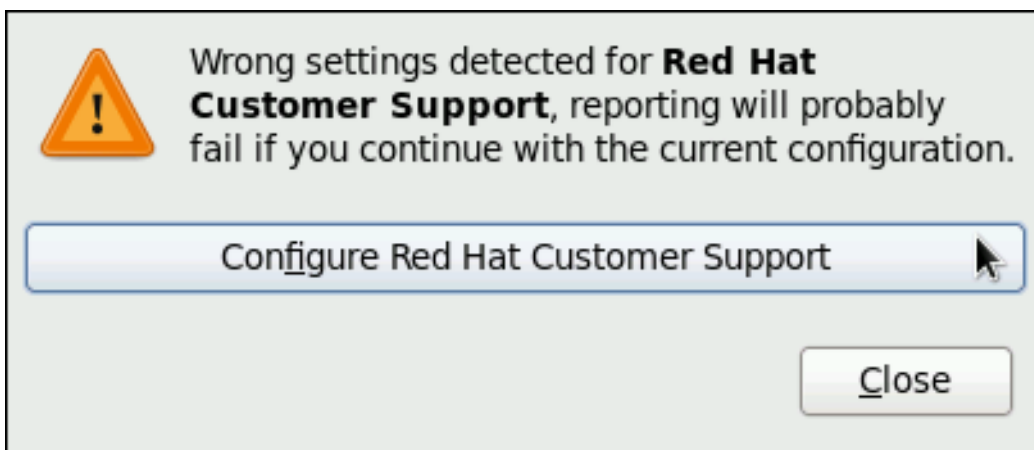
Figure 28.5. ABRT analyzing the problem

Next, choose how you want to report the issue. If you are using Red Hat Enterprise Linux, Red Hat Customer Support is the preferred choice.



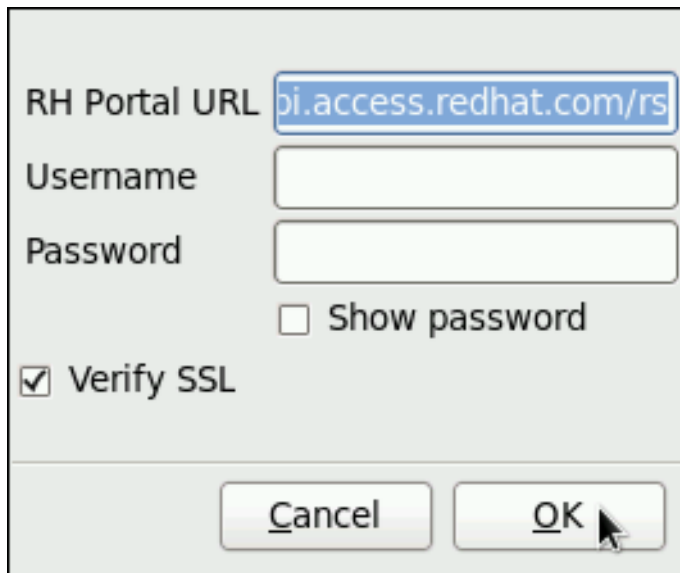
**Figure 28.6. Selecting a problem reporter**

If you choose to report to Red Hat Customer Support, and you have not configured this event yet, you will be warned that this event is not configured properly and you will be offered an option to do so.



**Figure 28.7. Warning - missing Red Hat Customer Support configuration**

Here, you need to provide your Red Hat login information (See [Section 28.4.3, “Event Configuration in ABRT GUI”](#) for more information on how to acquire it and how to set this event.), otherwise you will fail to report the problem.



RH Portal URL

Username

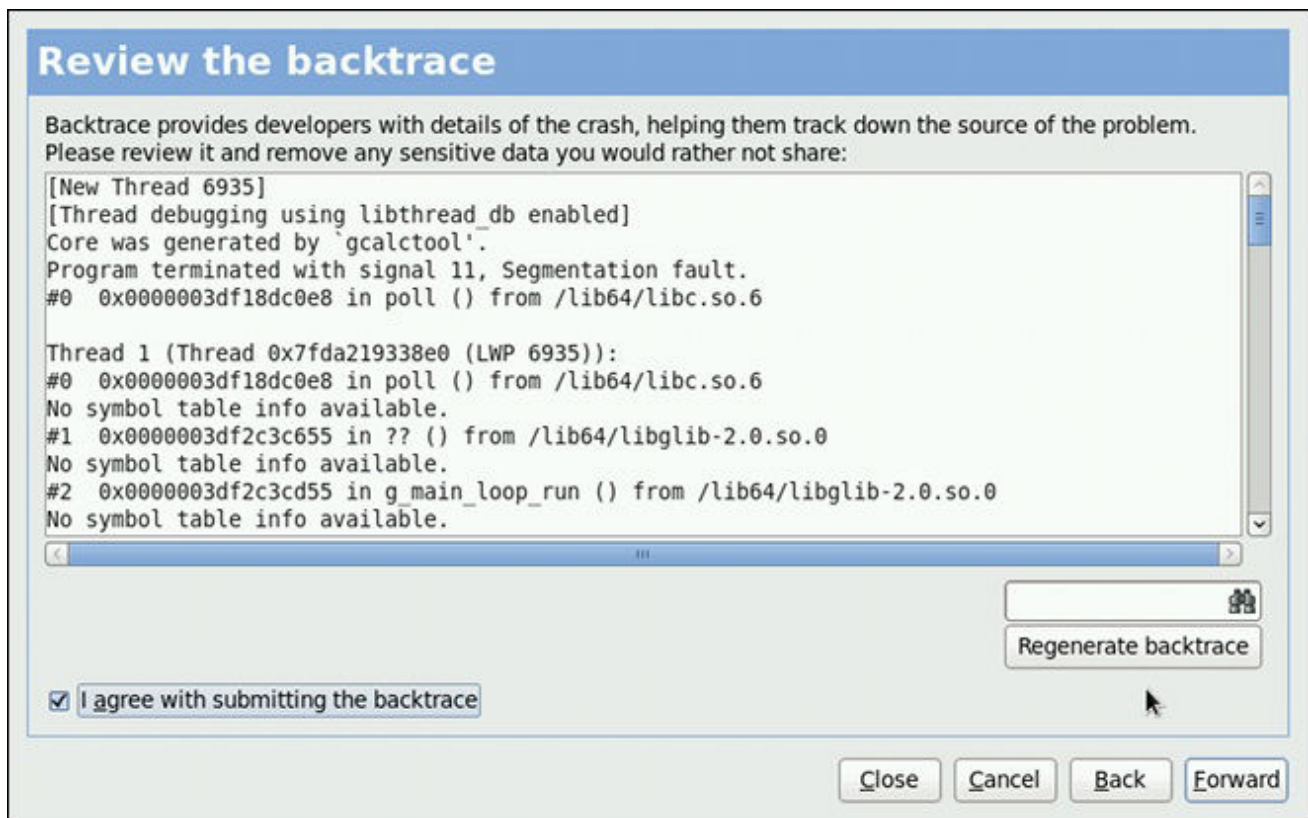
Password

Show password

Verify SSL

Figure 28.8. Red Hat Customer Support configuration window

After you have chosen a reporting method and have it set up correctly, review the backtrace and confirm the data to be reported.



**Review the backtrace**

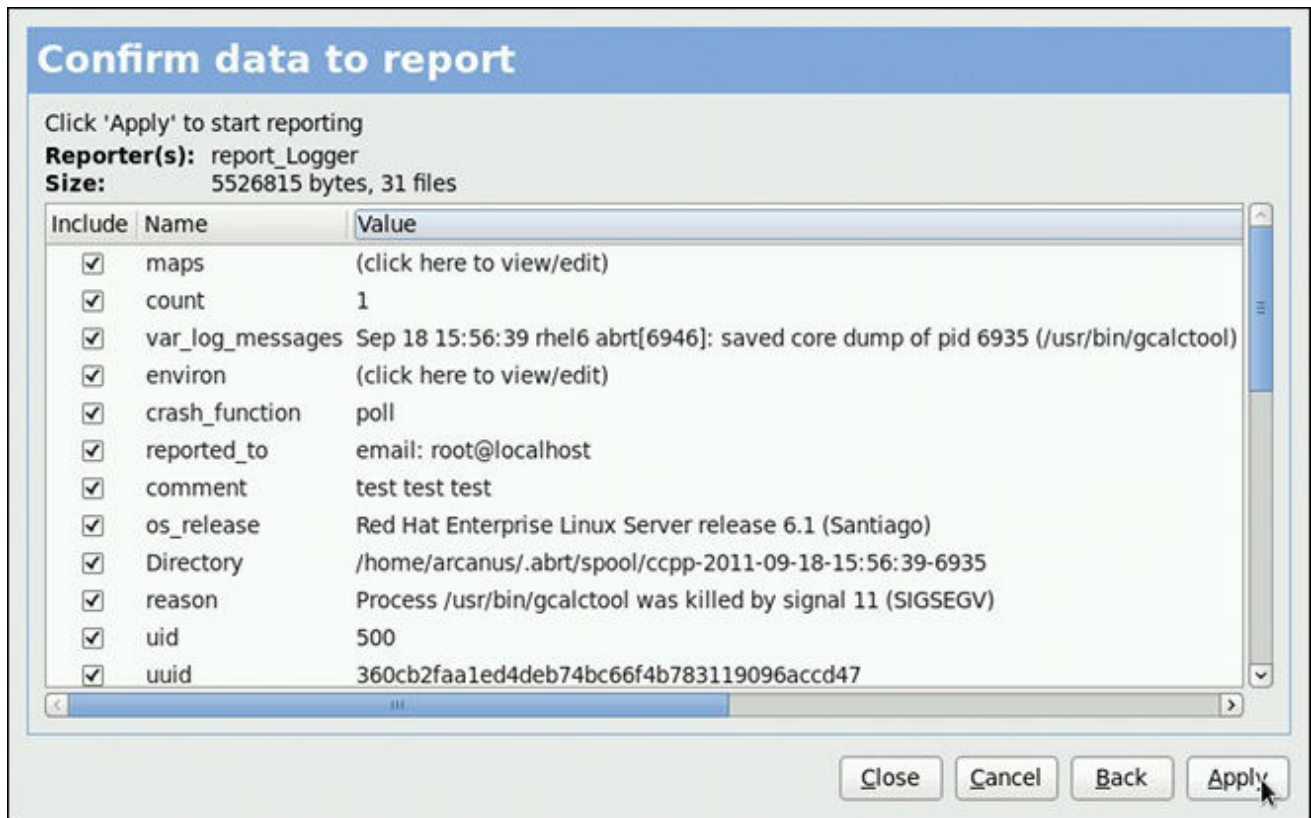
Backtrace provides developers with details of the crash, helping them track down the source of the problem. Please review it and remove any sensitive data you would rather not share:

```
[New Thread 6935]
[Thread debugging using libthread_db enabled]
Core was generated by `gcalctool'.
Program terminated with signal 11, Segmentation fault.
#0  0x0000003df18dc0e8 in poll () from /lib64/libc.so.6

Thread 1 (Thread 0x7fda219338e0 (LWP 6935)):
#0  0x0000003df18dc0e8 in poll () from /lib64/libc.so.6
No symbol table info available.
#1  0x0000003df2c3c655 in ?? () from /lib64/libglib-2.0.so.0
No symbol table info available.
#2  0x0000003df2c3cd55 in g_main_loop_run () from /lib64/libglib-2.0.so.0
No symbol table info available.
```

I agree with submitting the backtrace

Figure 28.9. Reviewing the problem backtrace



**Figure 28.10. Confirming the data to report**

Finally, the problem data is sent to the chosen destination, and you can now decide whether to continue with reporting the problem using another available method or finish your work on this problem. 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 by **ABRT** GUI when the problem case is created, or via emails received from Red Hat Support.

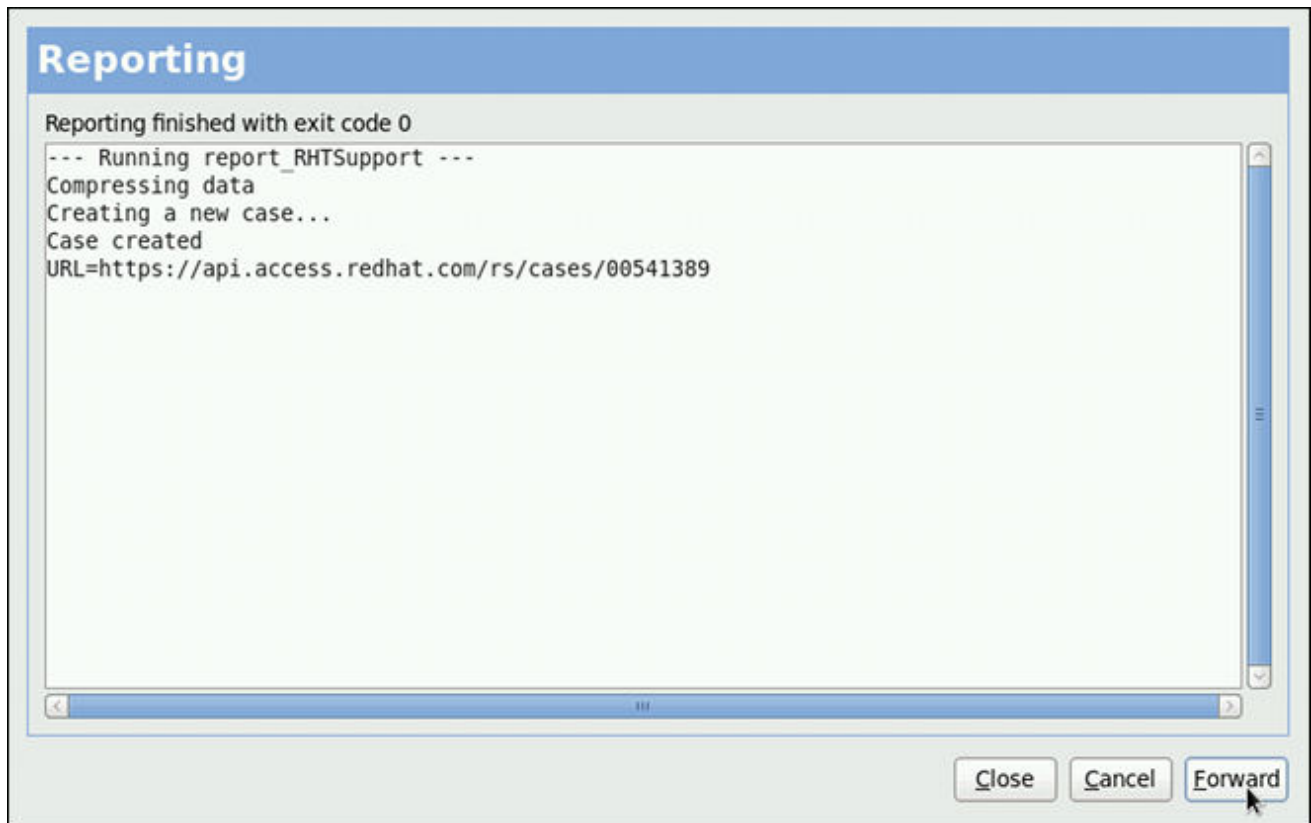


Figure 28.11. Problem is being reported to the Red Hat Customer Support database

## 28.3. USING THE COMMAND-LINE INTERFACE

Problem data saved by **abrt-d** can be viewed, reported, and deleted using the command-line interface.

General usage of the **abrt-cli** tool can be described using the following syntax:

```
abrt-cli [--version] command [args]
```

...where *args* stands for a problem data directory and/or options modifying the commands, and *command* is one of the following sub-commands:

- **list** — lists problems and views the problem data.
- **report** — analyzes and reports problems.
- **rm** — removes unneeded problems.
- **info** — provides information about a particular problem.

To display help on particular **abrt-cli** command use:

```
abrt-cli command --help
```

The rest of the commands used with **abrt-cli** are described in the following sections.

### 28.3.1. Viewing Problems

To view detected problems, enter the **abrt-cli list** command:

```

~]# abrt-cli list
Directory:      /var/spool/abrt/ccpp-2011-09-13-10:18:14-2895
count:         2
executable:    /usr/bin/gdb
package:       gdb-7.2-48.e16
time:         Tue 13 Sep 2011 10:18:14 AM CEST
uid:          500

Directory:      /var/spool/abrt/ccpp-2011-09-21-18:18:07-2841
count:         1
executable:    /bin/bash
package:       bash-4.1.2-8.e16
time:         Wed 21 Sep 2011 06:18:07 PM CEST
uid:          500

```

- **Directory** — Shows the problem data directory that contains all information about the problem.
- **count** — Shows how many times this particular problem occurred.
- **executable** — Indicates which binary or executable script crashed.
- **package** — Shows the name of the package that contains the program that caused the problem.
- **time** — Shows the date and time of the last occurrence of the problem.
- **uid** — Shows the ID of the user which ran the program that crashed.

The following table shows options available with the **abrt-cli list** command. All options are mutually inclusive so you can combine them according to your need. The command output will be the most comprehensive if you combine all options, and you will receive the least details if you use no additional options.

**Table 28.2. The abrt-cli list command options**

Option	Description
	With no additional option, the <b>abrt-cli list</b> command displays only basic information for problems that have not been reported yet.
<b>-d, --detailed</b>	Displays all stored information about problems listed, including a <b>backtrace</b> if it has already been generated.
<b>-f, --full</b>	Displays basic information for all problems including the already-reported ones.
<b>-v, --verbose</b>	Provides additional information on its actions.

If you want to view information just about one particular problem, you can use the command:

```
abrt-cli info directory
```

...where *directory* stands for the **problem data directory** of the problem that is being viewed. The following table shows options available with the **abrt-cli info** command. All options are mutually inclusive so you can combine them according to your need. The command output will be the most comprehensive if you combine all options, and you will receive the least details if you use no additional options.

**Table 28.3. The `abrt-cli info` command options**

Option	Description
	With no additional option, the <b>abrt-cli info</b> command displays only basic information for the problem specified by the <b>problem data directory</b> argument.
<b>-d, --detailed</b>	Displays all stored information for the problem specified by the <b>problem data directory</b> argument, including a <b>backtrace</b> if it has already been generated.
<b>-v, --verbose</b>	<b>abrt-cli info</b> provides additional information on its actions.

### 28.3.2. Reporting Problems

To report a certain problem, use the command:

```
abrt-cli report directory
```

...where *directory* stands for the **problem data directory** of the problem that is being reported. For example:

```
~]$ abrt-cli report /var/spool/abrt/ccpp-2011-09-13-10:18:14-2895
How you would like to analyze the problem?
1) Collect .xsession-errors
2) Local GNU Debugger
Select analyzer: _
```

**ABRT** prompts you to select an analyzer event for the problem that is being reported. After selecting an event, the problem is analyzed. This can take a considerable amount of time. When the problem report is ready, **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, **vi** 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. You will be asked which of the configured **ABRT** reporter events you want to use to send the report.

How would you like to report the problem?

- 1) Logger
  - 2) Red Hat Customer Support
- Select reporter(s): \_

After selecting a reporting method, you can proceed with reviewing data to be sent with the report. The following table shows options available with the **abrt-cli report** command.

**Table 28.4. The `abrt-cli report` command options**

Option	Description
	With no additional option, the <b>abrt-cli report</b> command provides the usual output.
<b>-v, --verbose</b>	<b>abrt-cli report</b> provides additional information on its actions.

### 28.3.3. Deleting Problems

If you are certain that you do not want to report a particular problem, you can delete it. To delete a problem so **ABRT** does not keep information about it, use the command:

```
abrt-cli rm directory
```

...where *directory* stands for the problem data directory of the problem being deleted. For example:

```
~]$ abrt-cli rm /var/spool/abrt/ccpp-2011-09-12-18:37:24-4413
rm '/var/spool/abrt/ccpp-2011-09-12-18:37:24-4413'
```

#### NOTE

Note that **ABRT** performs a detection of 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.

The following table shows options available with the **abrt-cli rm** command.

**Table 28.5. The `abrt-cli rm` command options**

Option	Description
	With no additional option, the <b>abrt-cli rm</b> command removes the specified <b>problem data directory</b> with all its contents.



Option	Description
<code>-v, --verbose</code>	<code>abrt-cli rm</code> provides additional information on its actions.

## 28.4. 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 — a problem is reported to Bugzilla.

When a problem is detected and its defining data is stored, the problem is processed by running events on the problem's data directory. For more information on events and how to define one, see [Section 28.4.1, “ABRT Events”](#). Standard **ABRT** installation currently supports several default events that can be selected and used during problem reporting process. See [Section 28.4.2, “Standard ABRT Installation Supported Events”](#) to see the list of these events.

Upon installation, **ABRT** and **libreport** place their respective configuration files into the several directories on a system:

- `/etc/libreport/` — contains the `report_event.conf` main configuration file. More information about this configuration file can be found in [Section 28.4.1, “ABRT Events”](#).
- `/etc/libreport/events/` — holds files specifying the default setting of predefined events.
- `/etc/libreport/events.d/` — keeps configuration files defining events.
- `/etc/libreport/plugins/` — contains configuration files of programs that take part in events.
- `/etc/abrt/` — holds **ABRT** specific configuration files used to modify the behavior of **ABRT**'s services and programs. More information about certain specific configuration files can be found in [Section 28.4.4, “ABRT Specific Configuration”](#).
- `/etc/abrt/plugins/` — keeps configuration files used to override the default setting of **ABRT**'s services and programs. For more information on some specific configuration files see [Section 28.4.4, “ABRT Specific Configuration”](#).

### 28.4.1. ABRT 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 used by the main configuration file, `/etc/libreport/report_event.conf`.

The `/etc/libreport/report_event.conf` file consists of *include directives* and *rules*. Rules are typically stored in other configuration files in the `/etc/libreport/events.d/` directory. In the standard installation, the `/etc/libreport/report_event.conf` file contains only one include directive:

```
include events.d/*.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, 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`, or
- `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 host name of the machine and its kernel version on the standard output.

Here is an example of a yet more complex event which is actually one of the predefined events. It saves relevant lines from the `~/.xsession-errors` file to the problem report for any problem for which the **abrt-ccpp** services has been used to process that problem, and the crashed application has loaded any X11 libraries at the time of 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 hard-set. System administrators can add events according to their need. Currently, the following event names are provided with standard **ABRT** and **libreport** installation:

### post-create

This event is run by **abrt-d** on newly created problem data directories. When the **post-create** event is run, **abrt-d** checks whether the UUID identifier of the new problem data matches the UUID of any already existing problem directories. If such a problem directory exists, the new problem data is deleted.

**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** runs the GNU Debugger (**GDB**) utility on a core dump of an application and produces a backtrace of a program. You can view the list of analyze events and choose from it using **abrt-gui**.

**collect\_name\_suffix**

...where *name\_suffix* is the adjustable part of the event name. This event is used to collect additional information on a problem. You can view the list of collect events and choose from it using **abrt-gui**.

**report\_name\_suffix**

...where *name\_suffix* is the adjustable part of the event name. This event is used to report a problem. You can view the list of report events and choose from it using **abrt-gui**.

Additional information about events (such as their description, names and types of parameters which can be passed to them as environment variables, and other properties) is stored in the `/etc/libreport/events/event_name.xml` files. These files are used by **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.

## 28.4.2. Standard ABRT Installation Supported Events

Standard **ABRT** installation currently provides a number of default analyzing, collecting and reporting events. Some of these events are also configurable using the **ABRT** GUI application (for more information on event configuration using **ABRT** GUI, see [Section 28.4.3, “Event Configuration in ABRT GUI”](#)). **ABRT** GUI only shows the event’s unique part of the name which is more readable the user, instead of the complete event name. For example, the **analyze\_xsession\_errors** event is shown as **Collect .xsession-errors** in **ABRT** GUI. The following is a list of default analyzing, collecting and reporting events provided by the standard installation of **ABRT**:

**analyze\_VMcore — Analyze VM core**

Runs **GDB** (the GNU debugger) on problem data of an application and generates a **backtrace** of the kernel. It is defined in the `/etc/libreport/events.d/vmcore_event.conf` configuration file.

**analyze\_LocalGDB — Local GNU Debugger**

Runs **GDB** (the GNU debugger) on problem data of an application and generates a **backtrace** of a program. It is defined in the `/etc/libreport/events.d/ccpp_event.conf` configuration file.

**analyze\_xsession\_errors — Collect .xsession-errors**

Saves relevant lines from the `~/.xsession-errors` file to the problem report. It is defined in the `/etc/libreport/events.d/ccpp_event.conf` configuration file.

**report\_Logger — Logger**

Creates a problem report and saves it to a specified local file. It is defined in the `/etc/libreport/events.d/print_event.conf` configuration file.

**report\_RHTSupport — Red Hat Customer Support**

Reports problems to the Red Hat Technical Support system. This possibility is intended for users of Red Hat Enterprise Linux. It is defined in the `/etc/libreport/events.d/rhtsupport_event.conf` configuration file.

#### **report\_Mailx — Mailx**

Sends a problem report via the **Mailx** utility to a specified email address. It is defined in the `/etc/libreport/events.d/mailx_event.conf` configuration file.

#### **report\_Kerneloops — Kerneloops.org**

Sends a kernel problem to the oops tracker. It is defined in the `/etc/libreport/events.d/koops_event.conf` configuration file.

#### **report\_Uploader — Report uploader**

Uploads a tarball (.tar.gz) archive with problem data to the chosen destination using the **FTP** or the **SCP** protocol. It is defined in the `/etc/libreport/events.d/uploader_event.conf` configuration file.

### 28.4.3. Event Configuration in ABRT GUI

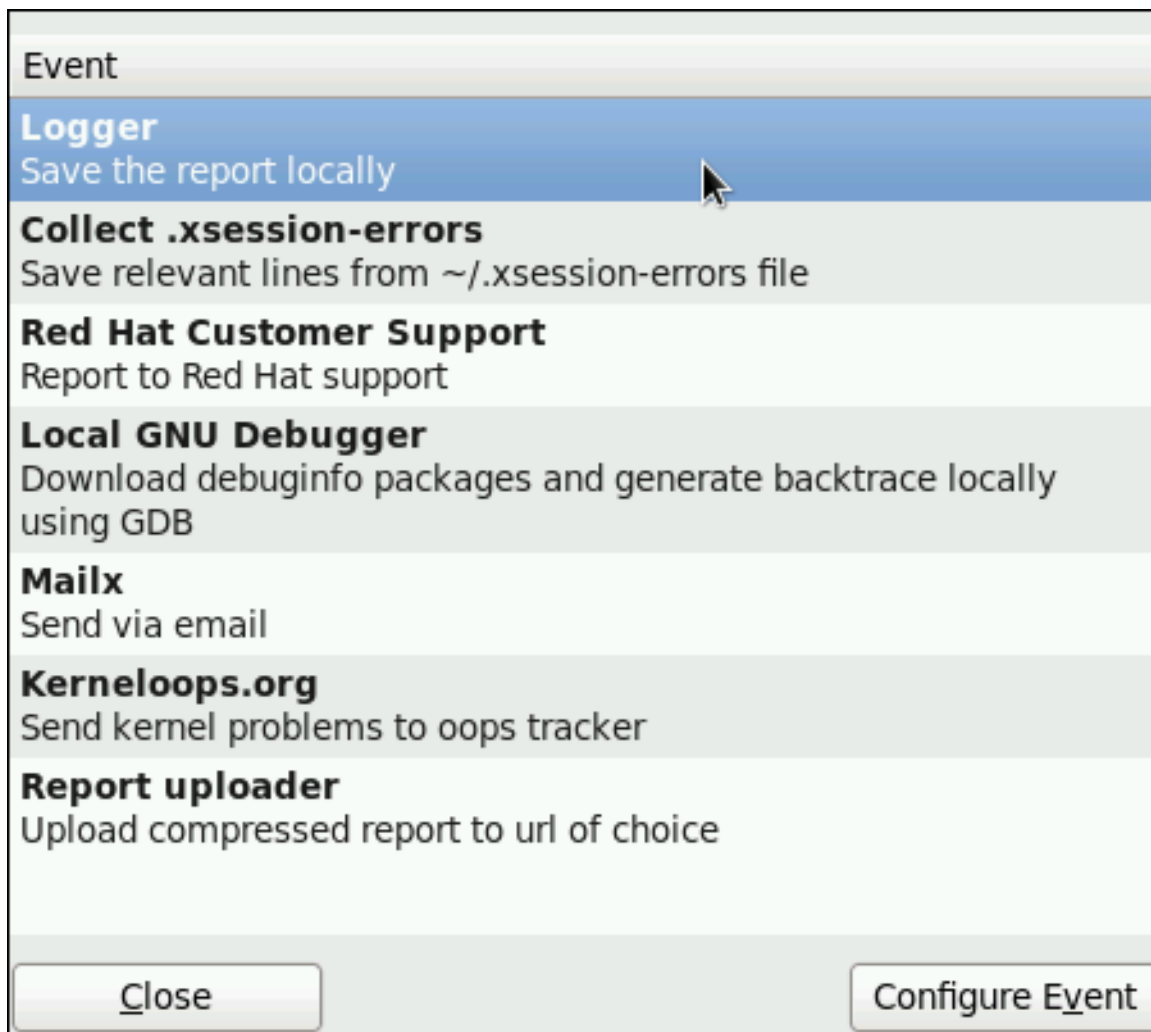
Events can use parameters passed to them as environment variables (for example, the **report\_Logger** event accepts an output file name as a parameter). Using the respective `/etc/libreport/events/event_name.xml` file, **ABRT** GUI determines which parameters can be specified for a selected event and allows a user to set the values for these parameters. These values are saved by **ABRT** GUI and reused on subsequent invocations of these events.

Open the **Event Configuration** window by clicking **Edit** → **Preferences**. This window shows a list of all available events that can be selected during the reporting process. When you select one of the configurable events, you can click the **Configure Event** button and you will be able to configure settings for that event. If you change any of the events' parameters, they are saved in the **Gnome** keyring and will be used in the future GUI sessions.



#### **NOTE**

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 user names, 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 stored in the **Gnome** keyring or can be stored in a text file in `$HOME/.abrt/*.conf` for use in **abrt-cli**.



**Figure 28.12. The Event Configuration Window**

The following is a list of all configuration options available for each predefined event that is configurable in the **ABRT** GUI application.

### Logger

In the **Logger** event configuration window, you can configure the following parameter:

- **Log file** — Specifies a file into which the crash reports are saved (by default, set to `/var/log/abrt.log`).

When the **Append** option is checked, the **Logger** event will append new crash reports to the log file specified in the **Logger file** option. When unchecked, the new crash report always replaces the previous one.

### Red Hat Customer Support

In the **Red Hat Customer Support** event configuration window, you can configure the following parameters:

- **RH Portal URL** — Specifies the Red Hat Customer Support URL where crash dumps are sent (by default, set to <https://api.access.redhat.com/rs>).
- **Username** — User login which is used to log into Red Hat Customer Support and create a Red Hat Customer Support database entry for a reported crash. Use your *Red Hat Login* acquired by creating an account on <https://www.redhat.com/en>, the Red Hat Customer Portal

(<https://access.redhat.com/home>) or the Red Hat Network (<https://rhn.redhat.com/>).

- **Password** — Password used to log into Red Hat Customer Support (that is, password associated with your *Red Hat Login*)

When the **SSL verify** option is checked, the **SSL** protocol is used when sending the data over the network.

## MailX

In the **MailX** event configuration window, you can configure the following parameters:

- **Subject** — A string that appears in the **Subject** field of a problem report email sent by **Mailx** (by default, set to "**[abrt] detected a crash**").
- **Sender** — A string that appears in the **From** field of a problem report email.
- **Recipient** — Email address of the recipient of a problem report email.

When the **Send Binary Data** option is checked, the problem report email will also contain all binary files associated with the problem in an attachment. The core dump file is also sent as an attachment.

## Kerneloops.org

In the **Kerneloops.org** event configuration window, you can configure the following parameter:

- **Kerneloops URL** — Specifies the URL where Kernel problems are reported to (by default, set to <http://submit.kerneloops.org/submitoops.php>)

## Report Uploader

In the **Report Uploader** event configuration window, you can configure the following parameter:

- **URL** — Specifies the URL where a tarball containing compressed problem data is uploaded using the **FTP** or **SCP** protocol (by default, set to **ftp://localhost:/tmp/upload**).

## 28.4.4. ABRT Specific Configuration

Standard **ABRT** installation currently provides the following **ABRT** specific configuration files:

- **/etc/abrt/abrt.conf** — allows you to modify the behavior of the **abrt** service.
- **/etc/abrt/abrt-action-save-package-data.conf** — allows you to modify the behavior of the **abrt-action-save-package-data** program.
- **/etc/abrt/plugins/CCpp.conf** — allows you to modify the behavior of **ABRT**'s core catching hook.

The following configuration directives are supported in the **/etc/abrt/abrt.conf** file:

### **WatchCrashdumpArchiveDir = /var/spool/abrt-upload**

This directive is commented out by default. Enable it if you want **abrt** to auto-unpack crashdump tarball archives (.tar.gz) which are located in the specified directory. In the example above, it is the **/var/spool/abrt-upload/** directory. Whichever directory you specify in this directive, you must

ensure that it exists and it is writable for **abrt**. The **ABRT** daemon will not create it automatically. If you change the default value of this option, be aware that in order to ensure proper functionality of **ABRT**, this directory **must not** be the same as the directory specified for the **DumpLocation** option.



### WARNING

Changing the location for crashdump archives will cause SELinux denials unless you reflect the change in respective SELinux rules first. See the **abrt\_selinux(8)** manual page for more information on running **ABRT** in SELinux.

Remember that if you enable this option when using SELinux, you need to execute the following command in order to set the appropriate Boolean allowing **ABRT** to write into the `public_content_rw_t` domain:

```
setsebool -P abrt_anon_write 1
```

### **MaxCrashReportsSize = *size\_in\_megabytes***

This option sets the amount of storage space, in megabytes, used by **ABRT** to store all problem information from all users. The default setting is **1000** MB. Once the quota specified here has been met, **ABRT** will continue catching problems, and in order to make room for the new crash dumps, it will delete the oldest and largest ones.

### **DumpLocation = `/var/spool/abrt`**

This directive is commented out by default. It specifies the location where problem data directories are created and in which problem core dumps and all other problem data are stored. The default location is set to the `/var/spool/abrt` directory. Whichever directory you specify in this directive, you must ensure that it exists and it is writable for **abrt**. If you change the default value of this option, be aware that in order to ensure proper functionality of **ABRT**, this directory **must not** be the same as the directory specified for the **WatchCrashdumpArchiveDir** option.



### WARNING

Changing the dump location will cause SELinux denials unless you reflect the change in respective SELinux rules first. See the **abrt\_selinux(8)** manual page for more information on running **ABRT** in SELinux.

Remember that if you enable this option when using SELinux, you need to execute the following command in order to set the appropriate Boolean allowing **ABRT** to write into the `public_content_rw_t` domain:

```
setsebool -P abrt_anon_write 1
```

The following configuration directives are supported in the `/etc/abrt/abrt-action-save-package-data.conf` file:

#### **OpenGPGCheck = yes/no**

Setting the **OpenGPGCheck** directive to **yes** (the default setting) tells **ABRT** to *only* analyze and handle crashes in applications provided by packages which are signed by the GPG keys whose locations are listed in the `/etc/abrt/gpg_keys` file. Setting **OpenGPGCheck** to **no** tells **ABRT** to catch crashes in all programs.

#### **BlackList = nspluginwrapper, valgrind, strace, [more\_packages]**

Crashes in packages and binaries listed after the **BlackList** directive will not be handled by **ABRT**. If you want **ABRT** to ignore other packages and binaries, list them here separated by commas.

#### **ProcessUnpackaged = yes/no**

This directive tells **ABRT** whether to process crashes in executables that do not belong to any package. The default setting is *no*.

#### **BlackListedPaths = /usr/share/doc/\*, \*/example\***

Crashes in executables in these paths will be ignored by **ABRT**.

The following configuration directives are supported in the `/etc/abrt/plugins/CCpp.conf` file:

#### **MakeCompatCore = yes/no**

This directive specifies whether **ABRT**'s core catching hook should create a core file, as it could be done if **ABRT** would not be installed. The core file is typically created in the current directory of the crashed program but only if the **ulimit -c** setting allows it. The directive is set to *yes* by default.

#### **SaveBinaryImage = yes/no**

This directive specifies whether **ABRT**'s core catching hook should save a binary image to a core dump. It is useful when debugging crashes which occurred in binaries that were deleted. The default setting is *no*.

### 28.4.5. Configuring ABRT to Detect a Kernel Panic

With Red Hat Enterprise Linux 6.3, **ABRT** can detect a kernel panic using the **abrt-vmcore** service, which is provided by the `abrt-addon-vmcore` package. The service starts automatically on system boot and searches for a core dump file in the `/var/crash/` directory. If a core dump file is found, **abrt-vmcore** creates the **problem data directory** in the `/var/spool/abrt/` directory and moves the core dump file to the newly created problem data directory. After the `/var/crash/` directory is searched through, the service is stopped until the next system boot.

To configure **ABRT** to detect a kernel panic, perform the following steps:

1. Ensure that the **kdump** service is enabled on the system. Especially, the amount of memory that is reserved for the `kdump` kernel has to be set correctly. You can set it by using the **system-config-kdump** graphical tool, or by specifying the **crashkernel** parameter in the list of kernel options in the `/etc/grub.conf` configuration file. See [Chapter 32, The kdump Crash Recovery Service](#) for details on how to enable and configure **kdump**.
2. Install the `abrt-addon-vmcore` package using the **Yum** package installer:



```
~]# yum install abrt-addon-vmcore
```

This installs the **abrt-vmcore** service with respective support and configuration files. Please note that the `abrt-addon-vmcore` package is provided by the Optional subscription channel. See [Section 8.4.8, “Adding the Optional and Supplementary Repositories”](#) for more information on Red Hat additional channels.

3. Reboot the system for the changes to take effect.

Unless **ABRT** is configured differently, problem data for any detected kernel panic is now stored in the `/var/spool/abrt/` directory and can be further processed by **ABRT** just as any other detected kernel oops.

### 28.4.6. Automatic Downloads and Installation of Debuginfo Packages

ABRT can be configured to automatically download and install packages needed for debugging of particular problems. This feature can be useful if you want to debug problems locally in your company environment. To enable automatic debuginfo downloads and installation, ensure that your system fulfills the following conditions:

- The `/etc/libreport/events.d/ccpp_event.conf` file contains the following analyzer event, which is present uncommented in default configuration:

```
EVENT=analyze_LocalGDB analyzer=CCpp
  abrt-action-analyze-core --core=coredump -o build_ids &&
  # In RHEL we don't want to install anything by default
  # and also this would fail, as the debuginfo repositories.
  # are not available without root password rhbz#759443
  # /usr/libexec/abrt-action-install-debuginfo-to-abrt-cache -
  -size_mb=4096 &&
  abrt-action-generate-backtrace &&
  abrt-action-analyze-backtrace
```

- The `/etc/libreport/events.d/ccpp_event.conf` file contains the following line, which allows ABRT to run binary to install debuginfo packages for the problems being analyzed. This line is, in order to avoid installations of unnecessary content, commented out by default so you have to remove the leading `#` character to enable it:

```
/usr/libexec/abrt-action-install-debuginfo-to-abrt-cache --
size_mb=4096 &&
```

- The `gdb` package, which allows you to generate a backtrace during a problem analysis, is installed on your system. If needed, see [Section 8.2.4, “Installing Packages”](#) for more information on how to install packages with the **Yum** package manager.



#### IMPORTANT

Note that debuginfo packages are installed using the **rhnpugin** which requires root privileges. Therefore, you have to run ABRT as **root** to be able to install debuginfo packages.

### 28.4.7. Configuring Automatic Reporting for Specific Types of Crashes

ABRT can be configured to report any detected issues or crashes automatically without any user interaction. This can be achieved by specifying an analyze-and-report rule as a *post-create* rule. For example, you can instruct ABRT to report Python crashes to Bugzilla immediately without any user interaction by enabling the rule and replacing the **EVENT=report\_Bugzilla** condition with the **EVENT=post-create** condition in the `/etc/libreport/events.d/python_event.conf` file. The new rule will look like the follows:

```
EVENT=post-create analyzer=Python
    test -f component || abrt-action-save-package-data
    reporter-bugzilla -c /etc/abrt/plugins/bugzilla.conf
```



### WARNING

Please note that the **post-create** event is run by **abrt-d**, which usually runs with root privileges.

#### 28.4.8. Uploading and Reporting Using a Proxy Server

The **reporter-bugzilla** and the **reporter-upload** tools respect the **http\_proxy** and the **ftp\_proxy** environment variables. When you use environment variables as a part of a reporting event, they inherit their values from the process which performs reporting, usually **abrt-gui** or **abrt-cli**. Therefore, you can specify **HTTP** or **FTP** proxy servers by using these variables in your working environment.

If you arrange these tools to be a part of the **post-create** event, they will run as children of the **abrt-d** process. You should either adjust the environment of **abrt-d** or modify the rules to set these variables. For example:

```
EVENT=post-create analyzer=Python
    test -f component || abrt-action-save-package-data
    export http_proxy=http://proxy.server:8888/
    reporter-bugzilla -c /etc/abrt/plugins/bugzilla.conf
```

#### 28.4.9. Configuring Automatic Reporting

ABRT can be configured to use *μReports*. This additional type of bug report has these advantages:

- Once enabled, *μReports* are sent *automatically*, without user interaction. In contrast, the normal reports are not sent until manually triggered by the user.
- *μReports* are *anonymous and do not contain sensitive information*. This eliminates the risk that unwanted data will be submitted automatically.
- A *μReport* represents the detected problem as a JSON object. Therefore, it is machine-readable and can be created and processed automatically.
- *μReports* are smaller than full bug reports.
- *μReports* do not require downloading large amounts of debugging information.

$\mu$ Reports serve several goals. They help to prevent duplicate customer cases that might get created because of multiple occurrences of the same bug. Additionally,  $\mu$ Reports enable gathering statistics of bug occurrences and finding known bugs across different systems. Finally, if *authenticated*  $\mu$ Reports are enabled as described at the end of this section, ABRT can automatically present instant solutions to the customers. However,  $\mu$ Reports do not necessarily provide engineers with enough information to fix the bug, for which a full bug report may be necessary.

A  $\mu$ Report generally contains the following information:

- a call stack trace of a program without any variables, or, in case of multi-threaded C, C++, and Java programs, multiple stack traces
- which operating system is used
- versions of the RPM packages involved in the crash
- whether the program ran under the **root** user
- for kernel oops, possibly information about host hardware



#### WARNING

Do not enable  $\mu$ Reports if you do not want to share information about your hardware with Red Hat.

For  $\mu$ Report examples, see the [Examples of  \$\mu\$ Reports](#) article.

With  $\mu$ Reports enabled, the following happens by default when a crash is detected:

1. ABRT submits a  $\mu$ Report with basic information about the problem to Red Hat's ABRT server.
2. The server determines whether the problem is already in the bug database.
3. If it is, the server returns a short description of the problem along with a URL of the reported case.

If not, the server invites the user to submit a full problem report.

To enable  $\mu$ Reports for all users, run as **root**:

```
~]# abrt-auto-reporting enabled
```

or add the following line to the `/etc/abrt/abrt.conf` file:

```
AutoreportingEnabled = yes
```

User-specific configuration is located in the `$USER/.config/abrt/` directory. It overrides the system-wide configuration.

To apply the new configuration, restart the ABRT services by running:

-

```
~]# service abrt restart
```

The default autoreporting behavior - sending  $\mu$ Reports - can be changed. To do that, assign a different ABRT event to the **AutoreportingEvent** directive in the `/etc/abrt/abrt.conf` configuration file. See [Section 28.4.2, “Standard ABRT Installation Supported Events”](#) for an overview of the standard events.

In Red Hat Enterprise Linux 7.1 and later, customers can also send *authenticated*  $\mu$ Reports, which contain more information: hostname, machine-id (taken from the `/etc/machine-id` file), and RHN account number. The advantage of authenticated  $\mu$ Reports is that they go directly to the Red Hat Customer Portal, and not only to Red Hat's private crash-report server, as the regular  $\mu$ Reports do. This enables Red Hat to provide customers with instant solutions to crashes.

To turn the *authenticated* automatic reporting on, run the following command as **root**:

```
~]# abrt-auto-reporting enabled -u RHN_username
```

Replace *RHN\_username* with your Red Hat Network username. This command will ask for your password and save it in plain text into the `/etc/libreport/plugins/rhnsupport.conf` file.

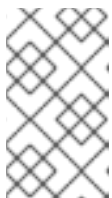
## 28.5. CONFIGURING CENTRALIZED CRASH COLLECTION

You can set up **ABRT** so that crash reports are collected from multiple systems and sent to a dedicated system for further processing. This is useful when an administrator does not want to log into hundreds of systems and manually check for crashes found by **ABRT**. In order to use this method, you need to install the **libreport-plugin-reportuploader** plug-in (`yum install libreport-plugin-reportuploader`). See the following sections on how to configure systems to use ABRT's centralized crash collection.

### 28.5.1. Configuration Steps Required on a Dedicated System

Complete the following steps on a dedicated (server) system:

1. Create a directory to which you want the crash reports to be uploaded to. Usually, `/var/spool/abrt-upload/` is used (the rest of the document assumes you are using this directory). Make sure this directory is writable by the `abrt` user.



#### NOTE

When the `abrt-desktop` package is installed, it creates a new system user and a group, both named **abrt**. This user is used by the `abrt` daemon, for example, as the owner:group of `/var/spool/abrt/*` directories.

2. In the `/etc/abrt/abrt.conf` configuration file, set the **WatchCrashdumpArchiveDir** directive to the following:

```
WatchCrashdumpArchiveDir = /var/spool/abrt-upload/
```

3. Choose your preferred upload mechanism; for example, **FTP** or **SCP**. For more information on how to configure **FTP**, see [Section 21.2, “FTP”](#). For more information on how to configure **SCP**, see [Section 14.4.2, “Using the scp Utility”](#).

It is advisable to check whether your upload method works. For example, if you use **FTP**, upload a file using an interactive **FTP** client:

```
~]$ ftp
ftp> open servername
Name: username
Password: password
ftp> cd /var/spool/abrt-upload
250 Operation successful
ftp> put testfile
ftp> quit
```

Check whether *testfile* appeared in the correct directory on the server system.

4. The **MaxCrashReportsSize** directive (in the `/etc/abrt/abrt.conf` configuration file) needs to be set to a larger value if the expected volume of crash data is larger than the default **1000 MB**.
5. Consider whether you would like to generate a backtrace of C/C++ crashes.

You can disable backtrace generation on the server if you do not want to generate backtraces at all, or if you decide to create them locally on the machine where a problem occurred. In the standard ABRT installation, a backtrace of a C/C++ crash is generated using the following rule in the `/etc/libreport/events.d/ccpp_events.conf` configuration file:

```
EVENT=analyze_LocalGDB analyzer=CCpp
    abrt-action-analyze-core.py --core=coredump -o build_ids &&
    abrt-action-install-debuginfo-to-abrt-cache --size_mb=4096
&&
    abrt-action-generate-backtrace &&
    abrt-action-analyze-backtrace
```

You can ensure that this rule is not applied for uploaded problem data by adding the **remote!=1** condition to the rule.

6. Decide whether you want to collect package information (the **package** and the **component** elements) in the problem data. See [Section 28.5.3, “Saving Package Information”](#) to find out whether you need to collect package information in your centralized crash collection configuration and how to configure it properly.

## 28.5.2. Configuration Steps Required on a Client System

Complete the following steps on every client system which will use the central management method:

1. If you do not want to generate a backtrace, or if you decided to generate it on a server system, you need to delete or comment out the corresponding rules in the `/etc/libreport/events.d/ccpp_events.conf` file. See [Section 28.5.1, “Configuration Steps Required on a Dedicated System”](#) for an example of such a example.
2. If you decided to not collect package information on client machines, delete, comment out or modify the rule which runs `abrt-action-save-package-data` in the `/etc/libreport/events.d/abrt_event.conf` file. See [Section 28.5.3, “Saving Package Information”](#) to find out whether you need to collect package information in your centralized crash collection configuration and how to configure it properly.

3. Add a rule for uploading problem reports to the server system in the corresponding configuration file. For example, if you want to upload all problems automatically as soon as they are detected, you can use the following rule in the `/etc/libreport/events.d/abrt_event.conf` configuration file:

```
EVENT=post-create
    reporter-upload -u scp://user:password@server.name/directory
```

Alternatively, you can use a similar rule that runs the `reporter-upload` program as the **report\_SFX** event if you want to store problem data locally on clients and upload it later using ABRT GUI/CLI. The following is an example of such an event:

```
EVENT=report_UploadToMyServer
    reporter-upload -u scp://user:password@server.name/directory
```

### 28.5.3. Saving Package Information

In a single-machine **ABRT** installation, problems are usually reported to external bug databases such as RHTSupport or Bugzilla. Reporting to these bug databases usually requires knowledge about the component and package in which the problem occurred. The **post-create** event runs the **abrt-action-save-package-data** tool (among other steps) in order to provide this information in the standard **ABRT** installation.

If you are setting up a centralized crash collection system, your requirements may be significantly different. Depending on your needs, you have two options:

#### Internal analysis of problems

After collecting problem data, you do not need to collect package information if you plan to analyze problems in-house, without reporting them to any external bug databases. You might be also interested in collecting crashes that occur in programs written by your organization or third-party applications installed on your system. If such a program is a part of an RPM package, then on *client systems* and a *dedicated crash collecting system*, you can only add the respective GPG key to the `/etc/abrt/gpg_keys` file or set the following line in the `/etc/abrt/abrt-action-save-package-data.conf` file:

```
OpenGPGCheck = no
```

If the program does not belong to any RPM package, take the following steps on both, *client systems* and a *dedicated crash collecting system*:

- Remove the following rule from the `/etc/libreport/events.d/abrt_event.conf` file:

```
EVENT=post-create component=
    abrt-action-save-package-data
```

- Prevent deletion of problem data directories which do not correspond to any installed package by setting the following directive in the `/etc/abrt/abrt-action-save-package-data.conf` file:

```
ProcessUnpackaged = yes
```

#### Reporting to external bug database

Alternatively, you may want to report crashes to RHTSupport or Bugzilla. In this case, you need to collect package information. Generally, client machines and dedicated crash collecting systems have non-identical sets of installed packages. Therefore, it may happen that problem data uploaded from a client does not correspond to any package installed on the dedicated crash collecting system. In the standard **ABRT** configuration, this will lead to deletion of problem data (ABRT will consider it to be a crash in an unpackaged executable). To prevent this from happening, it is necessary to modify **ABRT**'s configuration on the *dedicated system* in the following way:

- Prevent inadvertent collection of package information for problem data uploaded from client machines, by adding the **remote!=1** condition in the `/etc/libreport/events.d/abrt_event.conf` file:

```
EVENT=post-create remote!=1 component=
    abrt-action-save-package-data
```

- Prevent deletion of problem data directories which do not correspond to any installed package by setting the following directive in `/etc/abrt/abrt-action-save-package-data.conf`:

```
ProcessUnpackaged = yes
```



#### NOTE

Note that in this case, no such modifications are necessary on client systems: they continue to collect package information, and continue to ignore crashes in unpackaged executables.

### 28.5.4. Testing ABRT's Crash Detection

After completing all the steps of the configuration process, the basic setup is finished. To test that this setup works properly use the **kill -s SEGV PID** command to terminate a process on a client system. 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** should detect a crash shortly after executing the **kill** command. Check that the crash was detected by **ABRT** on the client system (this can be checked by examining the appropriate syslog file, by running the **abrt-cli list --full** command, or by examining the crash dump created in the `/var/spool/abrt` directory), copied to the server system, unpacked on the server system and can be seen and acted upon using **abrt-cli** or **abrt-gui** on the server system.

## CHAPTER 29. 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, essentially rolls over, 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.
- *oprofile does not associate samples for inline functions properly* — `oprofile` 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).

### 29.1. OVERVIEW OF TOOLS

Table 29.1, “OProfile Commands” provides a brief overview of the tools provided with the `oprofile` package.

**Table 29.1. OProfile Commands**



Command	Description
<b>ophelp</b>	Displays available events for the system's processor along with a brief description of each.
<b>opimport</b>	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.
<b>opannotate</b>	Creates annotated source for an executable if the application was compiled with debugging symbols. See <a href="#">Section 29.5.4, "Using opannotate"</a> for details.
<b>opcontrol</b>	Configures what data is collected. See <a href="#">Section 29.2, "Configuring OProfile"</a> for details.
<b>opreport</b>	Retrieves profile data. See <a href="#">Section 29.5.1, "Using opreport"</a> for details.
<b>oprofiled</b>	Runs as a daemon to periodically write sample data to disk.

## 29.2. CONFIGURING OPROFILE

Before OProfile can be run, 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.

### 29.2.1. 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
```



#### IMPORTANT

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 29.6, "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 29.2.3, “Separating Kernel and User-space Profiles”](#).

## 29.2.2. Setting Events to Monitor

Most processors contain *counters*, which are used by OProfile to monitor specific events. As shown in [Table 29.2, “OProfile Processors and Counters”](#), the number of counters available depends on the processor.

**Table 29.2. OProfile Processors and Counters**

Processor	cpu_type	Number of Counters
AMD64	x86-64/hammer	4
AMD Athlon	i386/athlon	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 S/390 and IBM System z	timer	1
Intel Core i7	i386/core_i7	4
Intel Nehalem microarchitecture	i386/nehalem	4
Intel Pentium 4 (non-hyper-threaded)	i386/p4	8
Intel Pentium 4 (hyper-threaded)	i386/p4-ht	4
Intel Westmere microarchitecture	i386/westmere	4

Processor	cpu_type	Number of Counters
TIMER_INT	timer	1

Use [Table 29.2, “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 29.3, “Default Events”](#).

**Table 29.3. 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
```



## NOTE

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 29.2, “Configuring OProfile”](#).

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 29.9, “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.

### 29.2.2.1. 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 whatever the jiffy rate is 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.



## WARNING

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.

### 29.2.2.2. 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
```

### 29.2.3. 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:kernel:0
```

Execute the following command to start profiling user mode for the counter again:

```
~]# opcontrol --event=event-name:sample-rate:unit-mask:kernel: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:

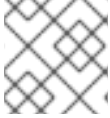
```
~]# opcontrol --separate=choice
```

*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.



## NOTE

These configuration changes will take effect when the OProfile profiler is restarted.

## 29.3. STARTING AND STOPPING OPROFILE

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**.



## IMPORTANT

On a Red Hat Enterprise Linux 6 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 to disable **nmi\_watchdog** at run time:

```
~]# echo 0 > /proc/sys/kernel/nmi_watchdog
```

To re-enable **nmi\_watchdog**, use the following command:

```
~]# echo 1 > /proc/sys/kernel/nmi_watchdog
```

To stop the profiler, execute the following command as root:

```
~]# opcontrol --shutdown
```

## 29.4. SAVING DATA

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.

## 29.5. ANALYZING THE DATA

Periodically, the OProfile daemon, **oprofiled**, 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.



### WARNING

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 29.4, “Saving Data”](#) for details on how to back up the sample file.

### 29.5.1. Using `opreport`

The `opreport` tool provides an overview of all the executables being profiled.

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 opprofiled
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.

### 29.5.2. Using `opreport` on a Single Executable

To retrieve more detailed profiled information about a specific executable, use `opreport`:

```
~]# opreport mode executable
```

*executable* must be the full path to the executable to be analyzed. *mode* must be one of the following:

-1



List sample data by symbols. For example, the following is part of the output from running the command **opreport -l /lib/tls/libc-version.so**:

```

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, the following output is from the command **opreport -l -i \_\_gconv\_transform\_utf8\_internal /lib/tls/libc-version.so**:

```

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**

List sample data by symbols with more detail than **-l**. For example, the following output is from the command **opreport -l -d \_\_gconv\_transform\_utf8\_internal /lib/tls/libc-version.so**:

```
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.

### **-x *symbol-name***

Exclude the comma-separated list of symbols from the output.

### **session:*name***

Specify the full path to the session or a directory relative to the `/var/lib/oprofile/samples/` directory.

## **29.5.3. 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 sample 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 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 --vmlinux=/usr/lib/debug/lib/modules/`uname -r`/vmlinux --event=CPU_CLK_UNHALTED:500000
```

Then the detailed information, for instance, for the `ext4` module can be obtained with:

```

~]# oprofile /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

```

#### 29.5.4. 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
```

The directory containing the source code and the executable to be analyzed must be specified. See the `opannotate` man page for a list of additional command-line options.

## 29.6. UNDERSTANDING `/DEV/OPROFILE/`

The `/dev/oprofile/` directory contains the file system for 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
```

## 29.7. 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 under performing, 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.

## 29.8. 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 6 includes built-in support for the JVM Tools Interface (JVMTI) agent library, which supports Java 1.5 and higher.

### 29.8.1. 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
```



#### NOTE

The `oprofile-jit` package must be installed on the system in order to profile JIT code with OProfile.

To learn more about Java support in OProfile, see the OProfile Manual, which is linked from [Section 29.11, “Additional Resources”](#).

## 29.9. 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.

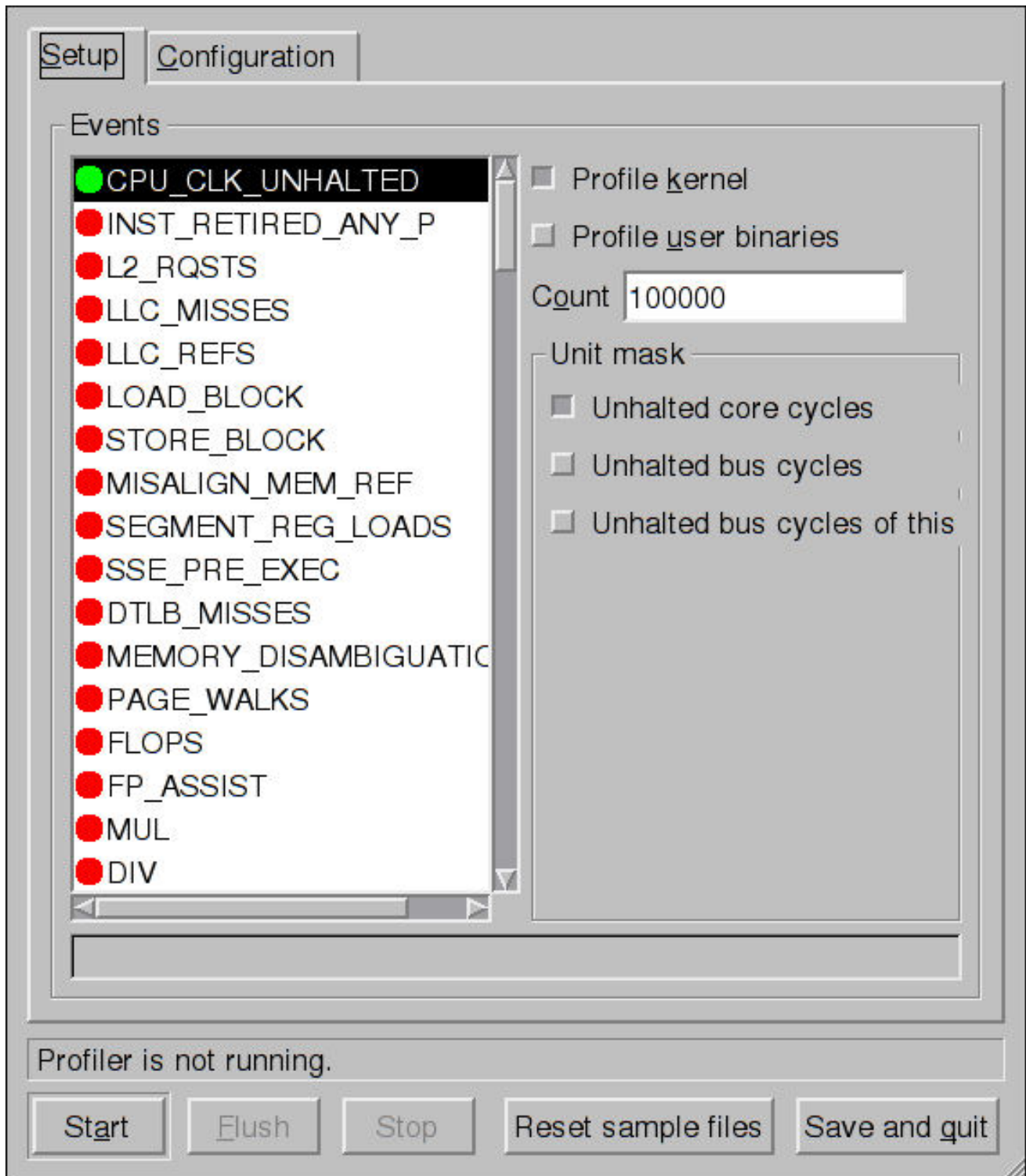
After changing any of the options, save them by clicking the **Save and quit** button. The preferences are written to `/root/.oprofile/daemonrc`, and the application exits.



### NOTE

Exiting the application does not stop OProfile from sampling.

On the **Setup** tab, to set events for the processor counters as discussed in [Section 29.2.2, “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 29.1. OProfile Setup**

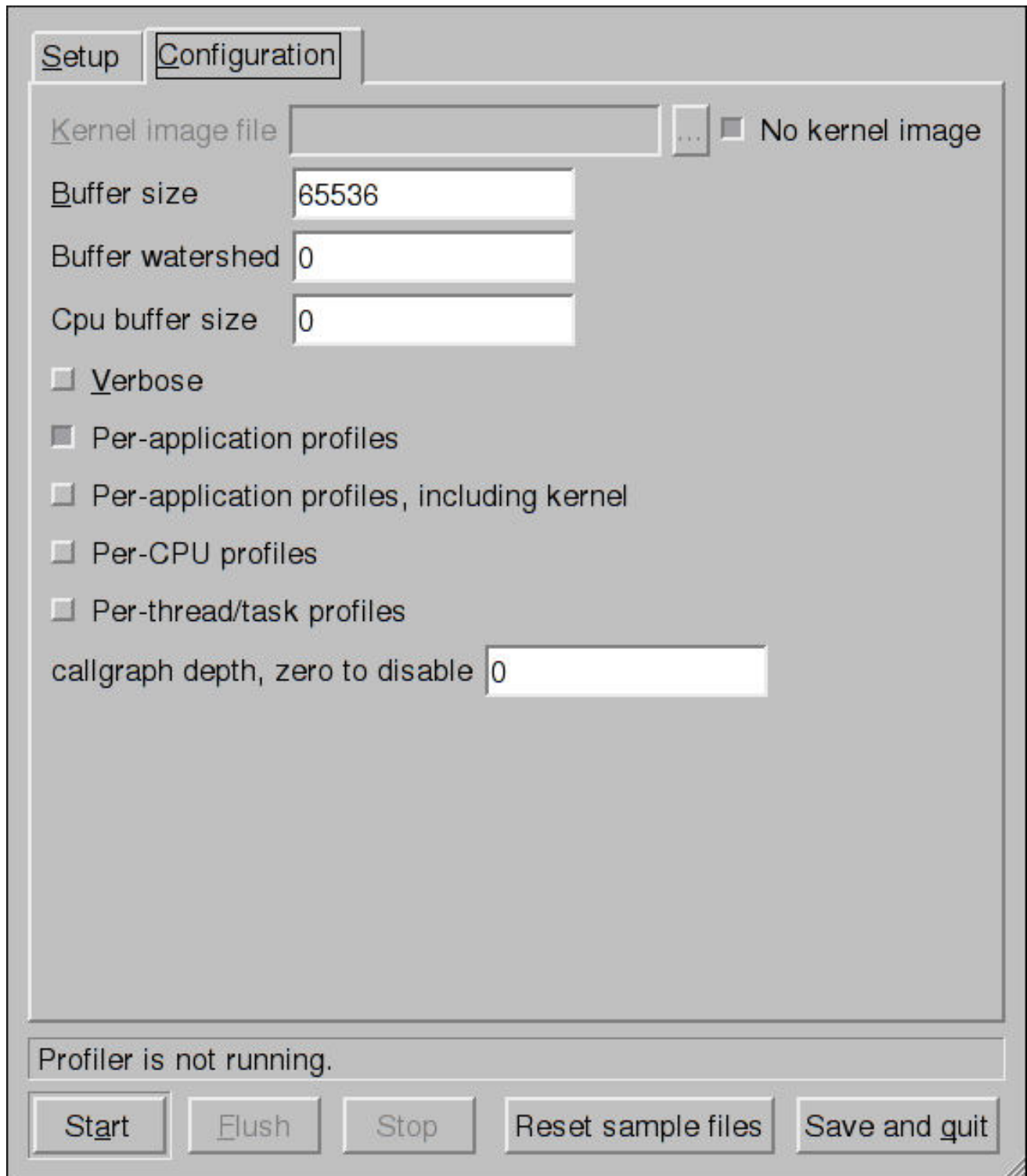
On the right side of the tab, select the **Profile kernel** option to count events in kernel mode for the currently selected event, as discussed in [Section 29.2.3, “Separating Kernel and User-space Profiles”](#). If this option is unselected, no samples are collected for the kernel.

Select the **Profile user binaries** option to count events in user mode for the currently selected event, as discussed in [Section 29.2.3, “Separating Kernel and User-space Profiles”](#). If this option is unselected, 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 29.2.2.1, “Sampling Rate”](#).

If any unit masks are available for the currently selected event, as discussed in [Section 29.2.2.2, “Unit Masks”](#), they are displayed in the **Unit Masks** area on the right side of the **Setup** tab. Select the check box 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 `vmlinux` 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 29.2. 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 29.2.3, “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 29.5, “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.

## 29.10. 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 instrumentation, it is particularly useful for instrumenting the kernel and daemons.

For more information on SystemTap, see [Section 29.11.2, “Useful Websites”](#) for the relevant SystemTap documentation.

## 29.11. ADDITIONAL RESOURCES

This chapter only highlights OProfile and how to configure and use it. To learn more, see the following resources.

### 29.11.1. Installed Docs

- `/usr/share/doc/oprofile-version/oprofile.html` — *OProfile Manual*
- `oprofile` man page — Discusses `opcontrol`, `opreport`, `opannotate`, and `ophelp`

### 29.11.2. 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 VIII. KERNEL, MODULE AND DRIVER CONFIGURATION**

This part covers various tools that assist administrators with kernel customization.

## CHAPTER 30. 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 8.1.2, “Updating Packages”](#). For information on Red Hat Network, see [Chapter 6, Registering the System and Managing Subscriptions](#).

### 30.1. 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 the development version of the kernel with numerous debugging options enabled for kernel diagnosis, at the expense of reduced performance.

- **kernel-doc** — Documentation files from the kernel source. Various portions of the Linux kernel and the device drivers shipped with it are documented in these files. Installation of this package provides a reference to the options that can be passed to Linux kernel modules at load time.

By default, these files are placed in the `/usr/share/doc/kernel-doc-<kernel_version>` directory.

- **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.
- **kernel-firmware** — Contains all of the firmware files that are required by various devices to operate.
- **perf** — This package contains supporting scripts and documentation for the **perf** tool shipped in each kernel image subpackage.

## 30.2. 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 **root**:

1. Install the **SYSLINUX** boot loader on the USB storage device:

```
~]# syslinux /dev/sdX1
```

...where *sdX* is the device name.

2. Create mount points for **boot.iso** and the USB storage device:

```
~]# mkdir /mnt/isoboot /mnt/diskboot
```

3. Mount **boot.iso**:

```
~]# mount -o loop boot.iso /mnt/isoboot
```

4. Mount the USB storage device:

```
~]# mount /dev/<sdX1> /mnt/diskboot
```

- Copy the **ISOLINUX** files from the **boot.iso** to the USB storage device:

```
~]# cp /mnt/isoboot/isolinux/* /mnt/diskboot
```

- 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
```

- Unmount **boot.iso** and the USB storage device:

```
~]# umount /mnt/isoboot /mnt/diskboot
```

- 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                2.6.32-17.el6           @rhel-x86_64-
server-6
kernel-doc.noarch           2.6.32-17.el6           @rhel-x86_64-
server-6
kernel-firmware.noarch      2.6.32-17.el6           @rhel-x86_64-
server-6
kernel-headers.x86_64       2.6.32-17.el6           @rhel-x86_64-
server-6
```

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 30.1, “Overview of Kernel Packages”](#) for descriptions of the different packages.

### 30.3. DOWNLOADING THE UPGRADED KERNEL

There are several ways to determine if an updated kernel is available for the system.

- Security Errata — See <http://www.redhat.com/security/updates/> for information on security errata, including kernel upgrades that fix security issues.
- The Red Hat Network — For a system subscribed to the Red Hat Network, the `yum` package manager can download the latest kernel and upgrade the kernel on the system. The `Dracut` utility will create an initial RAM disk image if needed, and configure the boot loader to boot the new kernel. For more information on installing packages from the Red Hat Network, see [Chapter 8, Yum](#). For more information on subscribing a system to the Red Hat Network, see [Chapter 6, Registering the System and Managing Subscriptions](#).

If **yum** was used to download and install the updated kernel from the Red Hat Network, follow the instructions in [Section 30.5, “Verifying the Initial RAM Disk Image”](#) and [Section 30.6, “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 30.4, “Performing the Upgrade”](#).

## 30.4. 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 30.5, “Verifying the Initial RAM Disk Image”](#) for details.

## 30.5. 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 6 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 [the section called “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 30.1. Verifying the Initial RAM Disk Image

1. As root, list the contents in the **/boot/** directory and find the kernel (**vmlinux-<kernel\_version>**) and **initramfs-<kernel\_version>** with the latest (most recent) version number:

#### Example 30.1. Ensuring that the kernel and initramfs versions match

```
~]# ls /boot/
config-2.6.32-17.el6.x86_64          lost+found
config-2.6.32-19.el6.x86_64          symvers-2.6.32-
17.el6.x86_64.gz
```

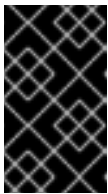
```

config-2.6.32-22.el6.x86_64          symvers-2.6.32-
19.el6.x86_64.gz                    symvers-2.6.32-
efi                                   symvers-2.6.32-
22.el6.x86_64.gz                     System.map-2.6.32-
grub                                  System.map-2.6.32-
17.el6.x86_64                         System.map-2.6.32-
initramfs-2.6.32-17.el6.x86_64.img   System.map-2.6.32-
19.el6.x86_64                         System.map-2.6.32-
initramfs-2.6.32-19.el6.x86_64.img   System.map-2.6.32-
22.el6.x86_64                         System.map-2.6.32-
initramfs-2.6.32-22.el6.x86_64.img   vmlinuz-2.6.32-17.el6.x86_64
initrd-2.6.32-17.el6.x86_64kdump.img vmlinuz-2.6.32-19.el6.x86_64
initrd-2.6.32-19.el6.x86_64kdump.img vmlinuz-2.6.32-22.el6.x86_64
initrd-2.6.32-22.el6.x86_64kdump.img

```

Example 30.1, “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-2.6.32-22.el6.x86_64`, and
- an `initramfs` file matching our kernel version, `initramfs-2.6.32-22.el6.x86_64.img`, also exists.



### IMPORTANT

In the `/boot/` directory you may find several `initrd-<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-2.6.32-
22.el6.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 initramfs-2.6.32-22.el6.x86_64.img
initramfs-2.6.32-22.el6.x86_64.img:
=====
====
dracut-004-17.el6
=====
====
drwxr-xr-x 23 root    root          0 May  3 22:34 .
drwxr-xr-x  2 root    root          0 May  3 22:33 proc
-rwxr-xr-x  1 root    root        7575 Mar 25 19:53 init
drwxr-xr-x  7 root    root          0 May  3 22:34 etc
drwxr-xr-x  2 root    root          0 May  3 22:34
etc/modprobe.d
[output truncated]
```

See `man dracut` and `man dracut.conf` for more information on options and usage.

3. Examine the `grub.conf` configuration file in the `/boot/grub/` directory to ensure that an `initrd initramfs-<kernel_version>.img` exists for the kernel version you are booting. See [Section 30.6, “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/vmlinitr-d-<kernel_version>` file already exists (the `<kernel_version>` should match the version of the kernel just installed).

## 30.6. 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.

In the following table, find your system's architecture to determine the boot loader it uses, and then click on the "See" link to jump to the correct instructions for your system.

**Table 30.1. Boot loaders by architecture**

Architecture	Boot Loader	See
--------------	-------------	-----

Architecture	Boot Loader	See
x86	GRUB	<a href="#">Section 30.6.1, “Configuring the GRUB Boot Loader”</a>
AMD AMD64 or Intel 64	GRUB	<a href="#">Section 30.6.1, “Configuring the GRUB Boot Loader”</a>
IBM eServer System i	OS/400	<a href="#">Section 30.6.3, “Configuring the OS/400 Boot Loader”</a>
IBM eServer System p	YABOOT	<a href="#">Section 30.6.4, “Configuring the YABOOT Boot Loader”</a>
IBM System z	z/IPL	

### 30.6.1. Configuring the GRUB Boot Loader

GRUB's configuration file, `/boot/grub/grub.conf`, contains a few lines with directives, such as **default**, **timeout**, **splashimage** and **hiddenmenu** (the last directive has no argument). The remainder of the file contains 4-line *stanzas* that each refer to an installed kernel. These stanzas always start with a **title** entry, after which the associated **root**, **kernel** and **initrd** directives should always be indented. Ensure that each stanza starts with a **title** that contains a version number (in parentheses) that matches the version number in the **kernel** `/vmlinuz-<version_number>` line of the same stanza.

#### Example 30.2. `/boot/grub/grub.conf`

```
# grub.conf generated by anaconda
[comments omitted]
default=1
timeout=0
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu

title Red Hat Enterprise Linux (2.6.32-22.el6.x86_64)
    root (hd0,0)
    kernel /vmlinuz-2.6.32-22.el6.x86_64 ro
root=/dev/mapper/vg_vm6b-lv_root rd_LVM_LV=vg_vm6b/lv_root rd_NO_LUKS
rd_NO_MD rd_NO_DM LANG=en_US.UTF-8 SYSFONT=latarcyrheb-sun16
KEYBOARDTYPE=pc KEYTABLE=us rhgb quiet crashkernel=auto
    initrd /initramfs-2.6.32-22.el6.x86_64.img

title Red Hat Enterprise Linux (2.6.32-19.el6.x86_64)
    root (hd0,0)
    kernel /vmlinuz-2.6.32-19.el6.x86_64 ro
root=/dev/mapper/vg_vm6b-lv_root rd_LVM_LV=vg_vm6b/lv_root rd_NO_LUKS
rd_NO_MD rd_NO_DM LANG=en_US.UTF-8 SYSFONT=latarcyrheb-sun16
KEYBOARDTYPE=pc KEYTABLE=us rhgb quiet crashkernel=auto
    initrd /initramfs-2.6.32-19.el6.x86_64.img

title Red Hat Enterprise Linux 6 (2.6.32-17.el6.x86_64)
```



```

    root (hd0,0)
    kernel /vmlinuz-2.6.32-17.el6.x86_64 ro
    root=/dev/mapper/vg_vm6b-lv_root rd_LVM_LV=vg_vm6b/lv_root rd_NO_LUKS
    rd_NO_MD rd_NO_DM LANG=en_US.UTF-8 SYSFONT=latarcyrheb-sun16
    KEYBOARDTYPE=pc KEYTABLE=us rhgb quiet
    initrd /initramfs-2.6.32-17.el6.x86_64.img

```

If a separate `/boot/` partition was created, the paths to the kernel and the `initramfs` image are relative to `/boot/`. This is the case in [Example 30.2, “/boot/grub/grub.conf”](#), above. Therefore the `initrd /initramfs-2.6.32-22.el6.x86_64.img` line in the first kernel stanza means that the `initramfs` image is actually located at `/boot/initramfs-2.6.32-22.el6.x86_64.img` when the root file system is mounted, and likewise for the kernel path (for example: `kernel /vmlinuz-2.6.32-22.el6.x86_64`) in each stanza of `grub.conf`.

## NOTE

In kernel boot stanzas in `grub.conf`, 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. Thus the `grub.conf` 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`

**Dracut** is a new utility available in Red Hat Enterprise Linux 6, and much-improved over `mkinitrd`. For information on using **Dracut**, see [Section 30.5, “Verifying the Initial RAM Disk Image”](#).

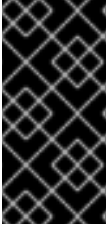
You should ensure that the kernel version number as given on the `kernel /vmlinuz-<kernel_version>` line matches the version number of the `initramfs` image given on the `initrd /initramfs-<kernel_version>.img` line of each stanza. See [Procedure 30.1, “Verifying the Initial RAM Disk Image”](#) for more information.

The `default=` directive tells GRUB which kernel to boot *by default*. Each `title` in `grub.conf` represents a bootable kernel. GRUB counts the `titled` stanzas representing bootable kernels starting with `0`. In [Example 30.2, “/boot/grub/grub.conf”](#), the line `default=1` indicates that GRUB will boot, by default, the *second* kernel entry, i.e. `title Red Hat Enterprise Linux (2.6.32-19.el6.x86_64)`.

In [Example 30.2, “/boot/grub/grub.conf”](#) GRUB is therefore configured to boot an older kernel, when we compare by version numbers. In order to boot the newer kernel, which is the *first* `title` entry in `grub.conf`, we would need to change the `default` value to `0`.

After installing a new kernel with `rpm`, verify that `/boot/grub/grub.conf` is correct, change the `default=` value to the new kernel (while remembering to count from `0`), and reboot the computer into the new kernel. Ensure your hardware is detected by watching the boot process output.

If GRUB presents an error and is unable to boot into the default kernel, it is often easiest to try to boot into an alternative or older kernel so that you can fix the problem.



## IMPORTANT

If you set the `timeout` directive in `grub.conf` to `0`, GRUB 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 will present you with the GRUB menu.

Alternatively, use the boot media you created earlier to boot the system.

### 30.6.2. Configuring the Loopback Device Limit

The maximum number of loopback devices in Red Hat Enterprise Linux 6 is set by the `max_loop` kernel option. For example, to set the maximum number of loopback devices to `64`, edit the `/etc/grub.conf` file, and add `max_loop=64` at the end of the kernel line. The line in `/etc/grub.conf` would then look something like this:

```
kernel /vmlinuz-2.6.32-131.0.15.el6.x86_64 ro root=/dev/mapper/root rhgb
quiet max_loop=64
initrd /initramfs-2.6.32-131.0.15.el6.x86_64.img
```

Reboot the system for the changes to take affect.

By default, eight `/dev/loop*` devices (`/dev/loop0` to `/dev/loop7`) are automatically generated, but others can be created as desired. For example, to set up a ninth loop device named `/dev/loop8`, issue the following command as `root`:

```
~]# mknod /dev/loop8 b 7 8
```

Thus, an administrator on a system with a Red Hat Enterprise Linux 6 kernel can create the desired number of loopback devices manually, with an init script, or with a `udev` rule.

However, if `max_loop` has been set before the system booted, `max_loop` becomes a hard limit on the number of loopback devices, and the number of loopback devices cannot be dynamically grown beyond the limit.

### 30.6.3. Configuring the OS/400 Boot Loader

The `/boot/vmlinitrd-<kernel-version>` file is installed when you upgrade the kernel. However, you must use the `dd` command to configure the system to boot the new kernel.

1. As root, issue the command `cat /proc/iSeries/mf/side` to determine the default side (either A, B, or C).
2. As root, issue the following command, where `<kernel-version>` is the version of the new kernel and `<side>` is the side from the previous command:

```
dd if=/boot/vmlinitrd-<kernel-version>
of=/proc/iSeries/mf/<side>/vmlinux bs=8k
```

Begin testing the new kernel by rebooting the computer and watching the messages to ensure that the hardware is detected properly.

### 30.6.4. Configuring the YABOOT Boot Loader

IBM eServer System p uses YABOOT as its boot loader. YABOOT uses `/etc/yaboot.conf` as its configuration file. Confirm that the file contains an **image** section with the same version as the kernel package just installed, and likewise for the **initramfs** image:

```
boot=/dev/sda1 init-message=Welcome to Red Hat Enterprise Linux! Hit <TAB>
for boot options
partition=2 timeout=30 install=/usr/lib/yaboot/yaboot delay=10 nonvram
image=/vmlinuz-2.6.32-17.EL
    label=old
    read-only
    initrd=/initramfs-2.6.32-17.EL.img
    append="root=LABEL=/"
image=/vmlinuz-2.6.32-19.EL
    label=linux
    read-only
    initrd=/initramfs-2.6.32-19.EL.img
    append="root=LABEL=/"
```

Notice that the default is not set to the new kernel. The kernel in the first image is booted by default. To change the default kernel to boot either move its image stanza so that it is the first one listed or add the directive **default** and set it to the **label** of the image stanza that contains the new kernel.

Begin testing the new kernel by rebooting the computer and watching the messages to ensure that the hardware is detected properly.

## CHAPTER 31. 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 `module-init-tools` 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 and/or unload modules manually, such as when a module provides optional functionality, one module should be preferred over another although either could provide basic functionality, or when a module is misbehaving, among other situations.

This chapter explains how to:

- use the user-space `module-init-tools` package 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 `module-init-tools` package is installed on your system by running, as root:

```
~]# yum install module-init-tools
```

For more information on installing packages with Yum, see [Section 8.2.4, “Installing Packages”](#).

### 31.1. 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
xfs                   803635  1
exportfs              3424   1 xfs
vfat                  8216   1
fat                   43410  1 vfat
tun                   13014  2
fuse                  54749  2
ip6table_filter       2743   0
```

```

ip6_tables           16558  1 ip6table_filter
ehtable_nat         1895  0
eatables            15186  1 ehtable_nat
ipt_MASQUERADE      2208  6
iptable_nat         5420  1
nf_nat              19059  2 ipt_MASQUERADE, iptable_nat
rfcomm              65122  4
ipv6                267017 33
sco                 16204  2
bridge              45753  0
stp                 1887  1 bridge
llc                 4557  2 bridge, stp
bnep                15121  2
l2cap               45185 16 rfcomm, bnep
cpufreq_ondemand    8420  2
acpi_cpufreq        7493  1
freq_table          3851  2 cpufreq_ondemand, acpi_cpufreq
usb_storage         44536  1
sha256_generic      10023  2
aes_x86_64          7654  5
aes_generic         27012  1 aes_x86_64
cbc                 2793  1
dm_crypt            10930  1
kvm_intel           40311  0
kvm                 253162 1 kvm_intel
[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 on the module you want to unload. For more information, see [Section 31.4, “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.

## 31.2. 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 module-init-tools 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 31.1. Listing information about a kernel module with lsmod**

```

~]# modinfo e1000e
filename:          /lib/modules/2.6.32-
71.el6.x86_64/kernel/drivers/net/e1000e/e1000e.ko
version:          1.2.7-k2
license:          GPL
description:      Intel(R) PRO/1000 Network Driver
author:           Intel Corporation, <linux.nics@intel.com>
srcversion:       93CB73D3995B501872B2982
alias:            pci:v00008086d00001503sv*sd*bc*sc*i*
alias:            pci:v00008086d00001502sv*sd*bc*sc*i*
[some alias lines omitted]
alias:            pci:v00008086d0000105Esv*sd*bc*sc*i*
depends:
vermagic:         2.6.32-71.el6.x86_64 SMP mod_unload modversions
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:             CrcStripping:Enable CRC Stripping, disable if your BMC
needs the CRC (array of int)
parm:             EEE:Enable/disable on parts that support the feature
(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**

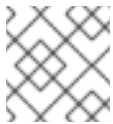
A short description of the module. You can use **modinfo -d** as a shortcut command for printing only the description field.

**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 31.2. 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:          EEE:Enable/disable on parts that support the feature
(array of 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)

```

## 31.3. LOADING A MODULE

To load a kernel module, run the **modprobe <module\_name>** command as root. For example, to load the **wacom** module, run:

```
~]# modprobe wacom
```

By default, **modprobe** attempts to load the module from the `/lib/modules/<kernel_version>/kernel/drivers/` directory. 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. A list of module dependencies is generated and maintained by the **depmod** program that is run automatically whenever a kernel or driver package is installed on the system. The **depmod** program keeps the list of dependencies in the `/lib/modules/<kernel_version>/modules.dep` file. The **modprobe** command always reads the **modules.dep** file 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: If necessary, it loads all dependencies of dependencies, and so on, 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 may include loading module dependencies. The following is an example of loading the **Fibre Channel over Ethernet** module verbosely:

### Example 31.3. modprobe -v shows module dependencies as they are loaded

```
~]# modprobe -v fcoe
insmod /lib/modules/2.6.32-71.el6.x86_64/kernel/drivers/scsi/scsi_tgt.ko
insmod /lib/modules/2.6.32-71.el6.x86_64/kernel/drivers/scsi/scsi_transport_fc.ko
insmod /lib/modules/2.6.32-71.el6.x86_64/kernel/drivers/scsi/libfc/libfc.ko
insmod /lib/modules/2.6.32-71.el6.x86_64/kernel/drivers/scsi/fcoe/libfcoe.ko
insmod /lib/modules/2.6.32-71.el6.x86_64/kernel/drivers/scsi/fcoe/fcoe.ko
```

This example 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.

## 31.4. 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 31.1, “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 (because you believe there is a bug in it that is affecting system stability, for example), your terminal session might look similar to this:

```
~]# modinfo -F depends firewire_ohci
depends:      firewire-core
~]# modinfo -F depends firewire_core
depends:      crc-itu-t
~]# modinfo -F depends crc-itu-t
depends:
```

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 /lib/modules/2.6.32-71.el6.x86_64/kernel/drivers/firewire/firewire-ohci.ko
rmmod /lib/modules/2.6.32-71.el6.x86_64/kernel/drivers/firewire/firewire-core.ko
rmmod /lib/modules/2.6.32-71.el6.x86_64/kernel/lib/crc-itu-t.ko
```

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.

## 31.5. BLACKLISTING A MODULE

Sometimes, for various performance or security reasons, it is necessary to prevent the system from using a certain kernel module. This can be achieved by *module blacklisting*, which is a mechanism used by the **modprobe** utility to ensure that the kernel cannot automatically load certain modules, or that the modules cannot be loaded at all. This is useful in certain situations, such as when using a certain module poses a security risk to your system, or when the module controls the same hardware or service as another module, and loading both modules would cause the system, or its component, to become unstable or non-operational.

To blacklist a module, you have to add the following line to the specified configuration file in the `/etc/modprobe.d/` directory as root:

```
blacklist <module_name>
```

where `<module_name>` is the name of the module being blacklisted.

You can modify the `/etc/modprobe.d/blacklist.conf` file that already exists on the system by default. However, the preferred method is to create a separate configuration file, `/etc/modprobe.d/<module_name>.conf`, that will contain settings specific only to the given kernel module.

#### Example 31.4. An example of `/etc/modprobe.d/blacklist.conf`

```
#
# Listing a module here prevents the hotplug scripts from loading it.
# Usually that'd be so that some other driver will bind it instead,
# no matter which driver happens to get probed first. Sometimes user
# mode tools can also control driver binding.
#
# Syntax: see modprobe.conf(5).
#

# watchdog drivers
blacklist i8xx_tco

# framebuffer drivers
blacklist aty128fb
blacklist atyfb
blacklist radeonfb
blacklist i810fb
blacklist cirrusfb
blacklist intelfb
blacklist kyrofb
blacklist i2c-matroxfb
blacklist hgafb
blacklist nvidiafb
blacklist rivafb
blacklist savagefb
blacklist sstfb
blacklist neofb
blacklist tridentfb
blacklist tdfxfb
blacklist virgefb
blacklist vga16fb
blacklist viafb

# ISDN - see bugs 154799, 159068
blacklist hisax
blacklist hisax_fcpcipnp

# sound drivers
blacklist snd-pcsp
```

```
# I/O dynamic configuration support for s390x (bz #563228)
blacklist chsc_sch
```

The **blacklist <module\_name>** command, however, does not prevent the module from being loaded manually, or from being loaded as a dependency for another kernel module that is not blacklisted. To ensure that a module cannot be loaded on the system at all, modify the specified configuration file in the **/etc/modprobe.d/** directory as root with the following line:

```
install <module_name> /bin/true
```

where *<module\_name>* is the name of the blacklisted module.

### Example 31.5. Using module blacklisting as a temporary problem solution

Let's say that a flaw in the Linux kernel's PPP over L2TP module (**pppol2tp**) has been found, and this flaw could be misused to compromise your system. If your system does not require the **pppol2tp** module to function, you can follow this procedure to blacklist **pppol2tp** completely until this problem is fixed:

1. Verify whether **pppol2tp** is currently loaded in the kernel by running the following command:

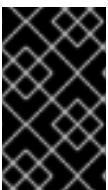
```
~]# lsmod | grep ^pppol2tp && echo "The module is loaded" || echo
"The module is not loaded"
```

2. If the module is loaded, you need to unload it and all its dependencies to prevent its possible misuse. See [Section 31.4, "Unloading a Module"](#) for instructions on how to safely unload it.
3. Run the following command to ensure that **pppol2tp** cannot be loaded to the kernel:

```
~]# echo "install pppol2tp /bin/true" >
/etc/modprobe.d/pppol2tp.conf
```

Note that this command overwrites the content of the **/etc/modprobe.d/pppol2tp.conf** file if it already exists on your system. Check and back up your existing **pppol2tp.conf** before running this command. Also, if you were unable to unload the module, you have to reboot the system for this command to take effect.

After the problem with the **pppol2tp** module has been properly fixed, you can delete the **/etc/modprobe.d/pppol2tp.conf** file or restore its previous content, which will allow your system to load the **pppol2tp** module with its original configuration.



#### IMPORTANT

Before blacklisting a kernel module, always ensure that the module is not vital for your current system configuration to function properly. Improper blacklisting of a key kernel module can result in an unstable or non-operational system.

## 31.6. 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. Load a kernel module by running the **modprobe** command along with a list of customized parameters on the command line. If the module is already loaded, you need to first unload all its dependencies and the module itself using the **modprobe -r** command. This method allows you to run a kernel module with specific settings without making the changes persistent. See [Section 31.6.1, “Loading a Customized Module - Temporary Changes”](#) for more information.
2. Alternatively, specify a list of the customized parameters in an existing or newly-created file in the `/etc/modprobe.d/` directory. This method ensures that the module customization is persistent by setting the specified parameters accordingly each time the module is loaded, such as after every reboot or **modprobe** command. See [Section 31.6.2, “Loading a Customized Module - Persistent Changes”](#) for more information.

### 31.6.1. Loading a Customized Module - Temporary Changes

Sometimes it is useful or necessary to run a kernel module temporarily with specific settings. To load a kernel module with customized parameters for the current system session, or until the module is reloaded with different parameters, run **modprobe** in the following format as root:

```
~]# modprobe <module_name> [parameter=value]
```

where `[parameter=value]` represents a list of customized parameters available to that module. 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.

The following procedure illustrates the recommended steps to load a kernel module with custom parameters on the **e1000e** module, which is the network driver for Intel PRO/1000 network adapters, as an example:

#### Procedure 31.1. Loading a Kernel Module with Custom Parameters

1. Verify whether the module is not already loaded into the kernel by running the following command:

```
~]# lsmod|grep e1000e
e1000e                236338  0
ptp                   9614    1 e1000e
```

Note that the output of the command in this example indicates that the **e1000e** module is already loaded into the kernel. It also shows that this module has one dependency, the **ptp** module.

2. If the module is already loaded into the kernel, you must unload the module and all its dependencies before proceeding with the next step. See [Section 31.4, “Unloading a Module”](#) for instructions on how to safely unload it.
3. 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 Energy Efficient Ethernet (EEE) turned on <sup>[5]</sup>, you would run, as root:

```
~]# modprobe e1000e InterruptThrottleRate=3000,3000,3000 EEE=1
```

This example illustrates passing multiple values to a single parameter by separating them with commas and omitting any spaces between them.

### 31.6.2. Loading a Customized Module - Persistent Changes

If you want to ensure that a kernel module is always loaded with specific settings, modify an existing or newly-created file in the `/etc/modprobe.d/` directory with a line in the following format.

```
~]# options <module_name> [parameter=value]
```

where `[parameter=value]` represents a list of customized parameters available to that module.

The following procedure illustrates the recommended steps for loading a kernel module with custom parameters on the **b43** module for Open Firmware for wireless networks, ensuring that changes persist between module reloads.

#### Procedure 31.2. Loading a Kernel Module with Custom Parameters - Persistent Changes

1. Add the following line to the `/etc/modprobe.d/openfwfw.conf` file, which ensures that the **b43** module is always loaded with QoS and hardware-accelerated cryptography disabled:

```
options b43 nohwcrypt=1 qos=0
```

2. Verify whether the module is not already loaded into the kernel by running the following command:

```
~]# lsmod | grep ^b43
~]#
```

Note that the output of the command in this example indicates that the module is currently not loaded into the kernel.

3. If the module is already loaded into the kernel, you must unload the module and all its dependencies before proceeding with the next step. See [Section 31.4, “Unloading a Module”](#) for instructions on how to safely unload it.
4. Load the **b43** module by running the following command:

```
~]# modprobe b43
```

## 31.7. PERSISTENT MODULE LOADING

As shown in [Example 31.1](#), “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 31.6. 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 31.7. `/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.

## 31.8. SPECIFIC KERNEL MODULE CAPABILITIES

This section explains how to enable specific kernel capabilities using various kernel modules.

### 31.8.1. Using Channel Bonding

Red Hat Enterprise Linux allows administrators to bind NICs together into a single channel using the `bonding` kernel module and a special network interface, called a *channel bonding interface*. Channel bonding enables two or more network interfaces to act as one, simultaneously increasing the bandwidth and providing redundancy.

To channel bond multiple network interfaces, the administrator must perform the following steps:

1. Configure a channel bonding interface as outlined in [Section 11.2.4, “Channel Bonding Interfaces”](#).
2. To enhance performance, adjust available module options to ascertain what combination works best. Pay particular attention to the `miimon` or `arp_interval` and the `arp_ip_target` parameters. See [Section 31.8.1.1, “Bonding Module Directives”](#) for a list of available options and how to quickly determine the best ones for your bonded interface.

### 31.8.1.1. Bonding Module Directives

It is a good idea to test which channel bonding module parameters work best for your bonded interfaces before adding them to the `BONDING_OPTS=<bonding parameters>` directive in your bonding interface configuration file (`ifcfg-bond0` for example). Parameters to bonded interfaces can be configured without unloading (and reloading) the bonding module by manipulating files in the `sysfs` file system.

`sysfs` is a virtual file system that represents kernel objects as directories, files and symbolic links. `sysfs` can be used to query for information about kernel objects, and can also manipulate those objects through the use of normal file system commands. The `sysfs` virtual file system has a line in `/etc/fstab`, and is mounted under the `/sys/` directory. All bonding interfaces can be configured dynamically by interacting with and manipulating files under the `/sys/class/net/` directory.

In order to determine the best parameters for your bonding interface, create a channel bonding interface file such as `ifcfg-bond0` by following the instructions in [Section 11.2.4, “Channel Bonding Interfaces”](#). Insert the `SLAVE=yes` and `MASTER=bond0` directives in the configuration files for each interface bonded to `bond0`. Once this is completed, you can proceed to testing the parameters.

First, bring up the bond you created by running `ifconfig bond<N> up` as root:

```
~]# ifconfig bond0 up
```

If you have correctly created the `ifcfg-bond0` bonding interface file, you will be able to see `bond0` listed in the output of running `ifconfig` (without any options):

```
~]# ifconfig
bond0    Link encap:Ethernet HWaddr 00:00:00:00:00:00
         UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
eth0    Link encap:Ethernet  HWaddr 52:54:00:26:9E:F1
         inet addr:192.168.122.251  Bcast:192.168.122.255
         Mask:255.255.255.0
         inet6 addr: fe80::5054:ff:fe26:9ef1/64 Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:207 errors:0 dropped:0 overruns:0 frame:0
         TX packets:205 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:70374 (68.7 KiB)  TX bytes:25298 (24.7 KiB)
[output truncated]
```

To view all existing bonds, even if they are not up, run:

```
~]# cat /sys/class/net/bonding_masters
bond0
```

You can configure each bond individually by manipulating the files located in the `/sys/class/net/bond<N>/bonding/` directory. First, the bond you are configuring must be taken down:

```
~]# ifconfig bond0 down
```

As an example, to enable MII monitoring on bond0 with a 1 second interval, you could run (as root):

```
~]# echo 1000 > /sys/class/net/bond0/bonding/miimon
```

To configure bond0 for **balance-alb** mode, you could run either:

```
~]# echo 6 > /sys/class/net/bond0/bonding/mode
```

...or, using the name of the mode:

```
~]# echo balance-alb > /sys/class/net/bond0/bonding/mode
```

After configuring options for the bond in question, you can bring it up and test it by running **ifconfig bond<N> up**. If you decide to change the options, take the interface down, modify its parameters using **sysfs**, bring it back up, and re-test.

Once you have determined the best set of parameters for your bond, add those parameters as a space-separated list to the **BONDING\_OPTS=** directive of the `/etc/sysconfig/network-scripts/ifcfg-bond<N>` file for the bonding interface you are configuring. Whenever that bond is brought up (for example, by the system during the boot sequence if the **ONBOOT=yes** directive is set), the bonding options specified in the **BONDING\_OPTS** will take effect for that bond. For more information on configuring bonding interfaces (and **BONDING\_OPTS**), see [Section 11.2.4, “Channel Bonding Interfaces”](#).

The following list provides the names of many of the more common channel bonding parameters, along with a descriptions of what they do. For more information, see the brief descriptions for each **parm** in **modinfo bonding** output, or the exhaustive descriptions in the **bonding.txt** file in the kernel-doc package (see [Section 31.9, “Additional Resources”](#)).

## Bonding Interface Parameters

### **arp\_interval=<time\_in\_milliseconds>**

Specifies (in milliseconds) how often ARP monitoring occurs. When configuring this setting, a good starting point for this parameter is **1000**.



### IMPORTANT

It is essential that both **arp\_interval** and **arp\_ip\_target** parameters are specified, or, alternatively, the **miimon** parameter is specified. Failure to do so can cause degradation of network performance in the event that a link fails.



If using this setting while in **mode=0** or **mode=2** (the two load-balancing modes), the network switch must be configured to distribute packets evenly across the NICs. For more information on how to accomplish this, see the **bonding.txt** file in the kernel-doc package (see [Section 31.9, “Additional Resources”](#)).

The value is set to **0** by default, which disables it.

**arp\_ip\_target=<ip\_address>[,<ip\_address\_2>,...<ip\_address\_16>]**

Specifies the target IP address of ARP requests when the **arp\_interval** parameter is enabled. Up to 16 IP addresses can be specified in a comma separated list.

**arp\_validate=<value>**

Validate source/distribution of ARP probes; default is **none**. Other valid values are **active**, **backup**, and **all**.

**downdelay=<time\_in\_milliseconds>**

Specifies (in milliseconds) how long to wait after link failure before disabling the link. The value must be a multiple of the value specified in the **miimon** parameter. The value is set to **0** by default, which disables it.

**lACP\_rate=<value>**

Specifies the rate at which link partners should transmit LACPDU packets in 802.3ad mode. Possible values are:

- **slow** or **0** — Default setting. This specifies that partners should transmit LACPDUs every 30 seconds.
- **fast** or **1** — Specifies that partners should transmit LACPDUs every 1 second.

**miimon=<time\_in\_milliseconds>**

Specifies (in milliseconds) how often MII link monitoring occurs. This is useful if high availability is required because MII is used to verify that the NIC is active. To verify that the driver for a particular NIC supports the MII tool, type the following command as root:

```
~]# ethtool <interface_name> | grep "Link detected:"
```

In this command, replace *<interface\_name>* with the name of the device interface, such as **eth0**, not the bond interface. If MII is supported, the command returns:

```
Link detected: yes
```

If using a bonded interface for high availability, the module for each NIC must support MII. Setting the value to **0** (the default), turns this feature off. When configuring this setting, a good starting point for this parameter is **100**.



## IMPORTANT

It is essential that both **arp\_interval** and **arp\_ip\_target** parameters are specified, or, alternatively, the **miimon** parameter is specified. Failure to do so can cause degradation of network performance in the event that a link fails.

**mode=<value>**

Allows you to specify the bonding policy. The <value> can be one of:

- **balance-rr** or **0** — Sets a round-robin policy for fault tolerance and load balancing. Transmissions are received and sent out sequentially on each bonded slave interface beginning with the first one available.
- **active-backup** or **1** — Sets an active-backup policy for fault tolerance. Transmissions are received and sent out via the first available bonded slave interface. Another bonded slave interface is only used if the active bonded slave interface fails.
- **balance-xor** or **2** — Sets an XOR (exclusive-or) policy for fault tolerance and load balancing. Using this method, the interface matches up the incoming request's MAC address with the MAC address for one of the slave NICs. Once this link is established, transmissions are sent out sequentially beginning with the first available interface.
- **broadcast** or **3** — Sets a broadcast policy for fault tolerance. All transmissions are sent on all slave interfaces.
- **802.3ad** or **4** — Sets an IEEE 802.3ad dynamic link aggregation policy. Creates aggregation groups that share the same speed and duplex settings. Transmits and receives on all slaves in the active aggregator. Requires a switch that is 802.3ad compliant.
- **balance-tlb** or **5** — Sets a Transmit Load Balancing (TLB) policy for fault tolerance and load balancing. The outgoing traffic is distributed according to the current load on each slave interface. Incoming traffic is received by the current slave. If the receiving slave fails, another slave takes over the MAC address of the failed slave. This mode is only suitable for local addresses known to the kernel bonding module and therefore cannot be used behind a bridge with virtual machines.
- **balance-alb** or **6** — Sets an Adaptive Load Balancing (ALB) policy for fault tolerance and load balancing. Includes transmit and receive load balancing for **IPv4** traffic. Receive load balancing is achieved through **ARP** negotiation. This mode is only suitable for local addresses known to the kernel bonding module and therefore cannot be used behind a bridge with virtual machines.

**num\_unsol\_na=<number>**

Specifies the number of unsolicited IPv6 Neighbor Advertisements to be issued after a failover event. One unsolicited NA is issued immediately after the failover.

The valid range is **0** - **255**; the default value is **1**. This parameter affects only the active-backup mode.

**primary=<interface\_name>**

Specifies the interface name, such as **eth0**, of the primary device. The **primary** device is the first of the bonding interfaces to be used and is not abandoned unless it fails. This setting is particularly useful when one NIC in the bonding interface is faster and, therefore, able to handle a bigger load.

This setting is only valid when the bonding interface is in **active-backup** mode. See the **bonding.txt** file in the kernel-doc package (see [Section 31.9, “Additional Resources”](#)).

**primary\_reselect=<value>**

Specifies the reselection policy for the primary slave. This affects how the primary slave is chosen to

become the active slave when failure of the active slave or recovery of the primary slave occurs. This parameter is designed to prevent flip-flopping between the primary slave and other slaves. Possible values are:

- **always** or **0** (default) — The primary slave becomes the active slave whenever it comes back up.
- **better** or **1** — The primary slave becomes the active slave when it comes back up, if the speed and duplex of the primary slave is better than the speed and duplex of the current active slave.
- **failure** or **2** — The primary slave becomes the active slave only if the current active slave fails and the primary slave is up.

The **primary\_reselect** setting is ignored in two cases:

- If no slaves are active, the first slave to recover is made the active slave.
- When initially enslaved, the primary slave is always made the active slave.

Changing the **primary\_reselect** policy via **sysfs** will cause an immediate selection of the best active slave according to the new policy. This may or may not result in a change of the active slave, depending upon the circumstances

#### **updelay=<time\_in\_milliseconds>**

Specifies (in milliseconds) how long to wait before enabling a link. The value must be a multiple of the value specified in the **miimon** parameter. The value is set to **0** by default, which disables it.

#### **use\_carrier=<number>**

Specifies whether or not **miimon** should use MII/ETHTOOL ioctls or **netif\_carrier\_ok()** to determine the link state. The **netif\_carrier\_ok()** function relies on the device driver to maintain its state with **netif\_carrier\_on/off**; most device drivers support this function.

The MII/ETHROOL ioctls tools utilize a deprecated calling sequence within the kernel. However, this is still configurable in case your device driver does not support **netif\_carrier\_on/off**.

Valid values are:

- **1** — Default setting. Enables the use of **netif\_carrier\_ok()**.
- **0** — Enables the use of MII/ETHTOOL ioctls.



#### **NOTE**

If the bonding interface insists that the link is up when it should not be, it is possible that your network device driver does not support **netif\_carrier\_on/off**.

#### **xmit\_hash\_policy=<value>**

Selects the transmit hash policy used for slave selection in **balance-xor** and **802.3ad** modes. Possible values are:

- **0** or **layer2** — Default setting. This parameter uses the XOR of hardware MAC addresses to generate the hash. The formula used is:

$$(| \quad (<source\_MAC\_address> \text{ XOR } <destination\_MAC>) \text{ MODULO } <slave\_count>$$

This algorithm will place all traffic to a particular network peer on the same slave, and is 802.3ad compliant.

- **1 or layer3+4** — Uses upper layer protocol information (when available) to generate the hash. This allows for traffic to a particular network peer to span multiple slaves, although a single connection will not span multiple slaves.

The formula for unfragmented TCP and UDP packets used is:

$$(| \quad ((<source\_port> \text{ XOR } <dest\_port>) \text{ XOR } \\ \quad ((<source\_IP> \text{ XOR } <dest\_IP>) \text{ AND } 0xffff) \\ \quad \text{MODULO } <slave\_count>$$

For fragmented TCP or UDP packets and all other IP protocol traffic, the source and destination port information is omitted. For non-IP traffic, the formula is the same as the **layer2** transmit hash policy.

This policy intends to mimic the behavior of certain switches; particularly, Cisco switches with PFC2 as well as some Foundry and IBM products.

The algorithm used by this policy is not 802.3ad compliant.

- **2 or layer2+3** — Uses a combination of layer2 and layer3 protocol information to generate the hash.

Uses XOR of hardware MAC addresses and IP addresses to generate the hash. The formula is:

$$(| \quad (((<source\_IP> \text{ XOR } <dest\_IP>) \text{ AND } 0xffff) \text{ XOR } \\ \quad ( <source\_MAC> \text{ XOR } <destination\_MAC> )) \\ \quad \text{MODULO } <slave\_count>$$

This algorithm will place all traffic to a particular network peer on the same slave. For non-IP traffic, the formula is the same as for the layer2 transmit hash policy.

This policy is intended to provide a more balanced distribution of traffic than layer2 alone, especially in environments where a layer3 gateway device is required to reach most destinations.

This algorithm is 802.3ad compliant.

## 31.9. ADDITIONAL RESOURCES

For more information on kernel modules and their utilities, see the following resources.

### Installed Documentation

- **lsmod(8)** — The manual page for the **lsmod** command.
- **modinfo(8)** — The manual page for the **modinfo** command.
- **modprobe(8)** — The manual page for the **modprobe** command.

- **rmmod(8)** — The manual page for the **rmmod** command.
- **ethtool(8)** — The manual page for the **ethtool** command.
- **mii-tool(8)** — The manual page for the **mii-tool** command.

## Installable Documentation

- **/usr/share/doc/kernel-doc-<kernel\_version>/Documentation/** — This directory, which is provided by the **kernel-doc** package, contains information on the kernel, kernel modules, and their respective parameters. Before accessing the kernel documentation, you must run the following command as root:

```
~]# yum install kernel-doc
```

## Online Documentation

- — [The Red Hat Knowledgebase article \*Which bonding modes work when used with a bridge that virtual machine guests connect to?\*](#)

---

[5] Despite what the example might imply, Energy Efficient Ethernet is turned on by default in the **e1000e** driver.

## CHAPTER 32. THE KDUMP CRASH RECOVERY SERVICE

When the **kdump** crash dumping mechanism is enabled, the system is booted from the context of another kernel. This second kernel reserves a small amount of memory and its only purpose is to capture the core dump image in case the system crashes.

Being able to analyze the core dump significantly helps to determine the exact cause of the system failure, and it is therefore strongly recommended to have this feature enabled. This chapter explains how to configure, test, and use the **kdump** service in Red Hat Enterprise Linux, and provides a brief overview of how to analyze the resulting core dump using the **crash** debugging utility.

### 32.1. INSTALLING THE KDUMP SERVICE

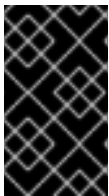
In order to use the **kdump** service on your system, make sure you have the **kexec-tools** package installed. To do so, type the following at a shell prompt as **root**:

```
~]# yum install kexec-tools
```

For more information on how to install new packages in Red Hat Enterprise Linux, see [Section 8.2.4, “Installing Packages”](#).

### 32.2. CONFIGURING THE KDUMP SERVICE

There are three common means of configuring the **kdump** service: at the first boot, using the **Kernel Dump Configuration** graphical utility, and doing so manually on the command line.



#### IMPORTANT

A limitation in the current implementation of the **Intel IOMMU** driver can occasionally prevent the **kdump** service from capturing the core dump image. To use **kdump** on Intel architectures reliably, it is advised that the IOMMU support is disabled.



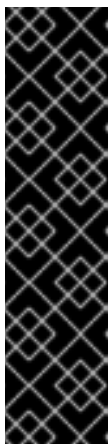
#### WARNING

It is known that the **kdump** service does not work reliably on certain combinations of HP Smart Array devices and system boards from the same vendor. Consequent to this, users are strongly advised to test the configuration before using it in production environment, and if necessary, configure **kdump** to store the kernel crash dump to a remote machine over a network. For more information on how to test the **kdump** configuration, see [Section 32.2.4, “Testing the Configuration”](#).

#### 32.2.1. Configuring kdump at First Boot

When the system boots for the first time, the **firstboot** application is launched to guide the user through the initial configuration of the freshly installed system. To configure **kdump**, navigate to the **Kdump** section and follow the instructions below.

1. Select the **Enable kdump?** check box to allow the **kdump** daemon to start at boot time. This will enable the service for runlevels **2**, **3**, **4**, and **5**, and start it for the current session. Similarly, unselecting the check box will disable it for all runlevels and stop the service immediately.
2. Click the up and down arrow buttons next to the **Kdump Memory** field to increase or decrease the value to configure the amount of memory that is reserved for the **kdump** kernel. Notice that the **Usable System Memory** field changes accordingly showing you the remaining memory that will be available to the system.



### IMPORTANT

This section is available only if the system has enough memory. To learn about minimum memory requirements of the Red Hat Enterprise Linux 6 system, read the *Required minimums* section of the [Red Hat Enterprise Linux Technology Capabilities and Limits](#) comparison chart. When the **kdump** crash recovery is enabled, the minimum memory requirements increase by the amount of memory reserved for it. This value is determined by the user, and defaults to 128 MB plus 64 MB for each TB of physical memory (that is, a total of 192 MB for a system with 1 TB of physical memory). The memory can be attempted up to the maximum of 896 MB if required. This is recommended especially in large environments, for example in systems with a large number of Logical Unit Numbers (LUNs).

## 32.2.2. Using the Kernel Dump Configuration Utility

To start the **Kernel Dump Configuration** utility, select **System** → **Administration** → **Kernel crash dumps** from the panel, or type `system-config-kdump` at a shell prompt. You will be presented with a window as shown in [Figure 32.1](#), “[Basic Settings](#)”.

The utility allows you to configure **kdump** as well as to enable or disable starting the service at boot time. When you are done, click **Apply** to save the changes. The system reboot will be requested, and unless you are already authenticated, you will be prompted to enter the superuser password.



### IMPORTANT

On IBM System z or PowerPC systems with SELinux running in Enforcing mode, the `kdumpgui_run_bootloader` Boolean must be enabled before launching the **Kernel Dump Configuration** utility. This Boolean allows `system-config-kdump` to run the boot loader in the `bootloader_t` SELinux domain. To permanently enable the Boolean, run the following command as **root**:

```
~]# setsebool -P kdumpgui_run_bootloader 1
```

### Enabling the Service

To start the **kdump** daemon at boot time, click the **Enable** button on the toolbar. This will enable the service for runlevels **2**, **3**, **4**, and **5**, and start it for the current session. Similarly, clicking the **Disable** button will disable it for all runlevels and stop the service immediately.

For more information on runlevels and configuring services in general, see [Chapter 12, Services and Daemons](#).

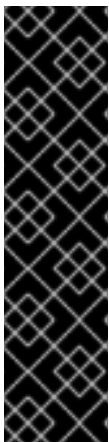
### The Basic Settings Tab

The **Basic Settings** tab enables you to configure the amount of memory that is reserved for the **kdump** kernel. To do so, select the **Manual kdump memory settings** radio button, and click the up

and down arrow buttons next to the **New kdump Memory** field to increase or decrease the value. Notice that the **Usable Memory** field changes accordingly showing you the remaining memory that will be available to the system.



Figure 32.1. Basic Settings



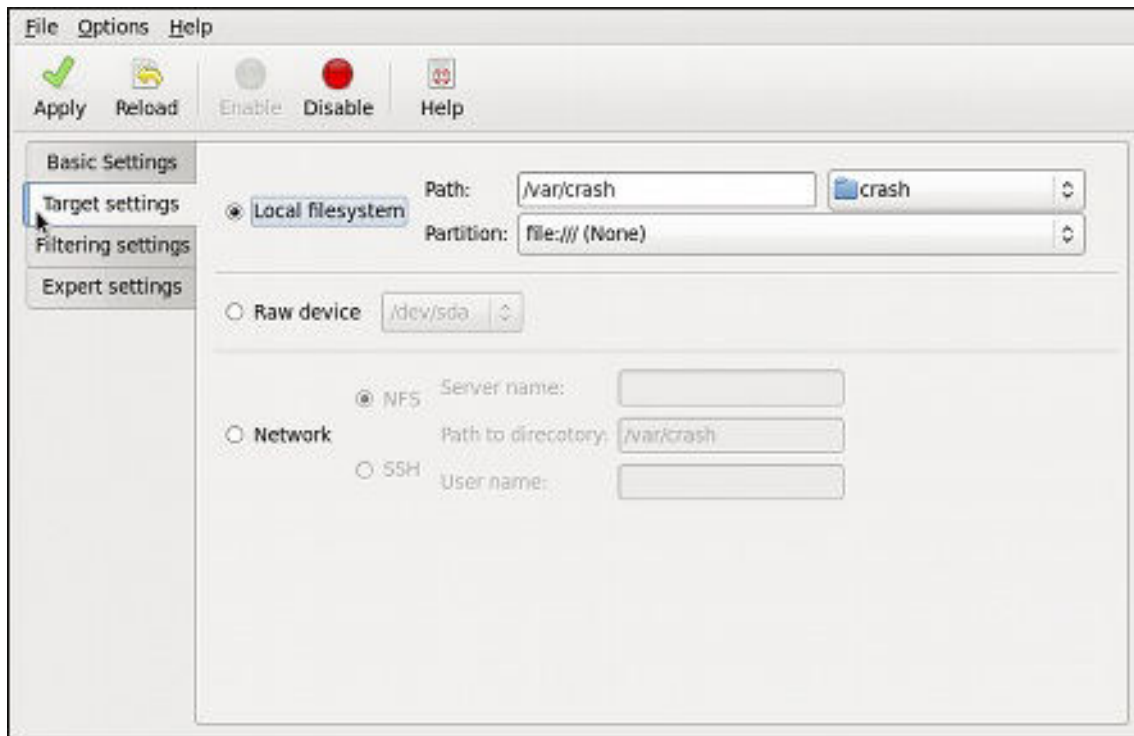
## IMPORTANT

This section is available only if the system has enough memory. To learn about minimum memory requirements of the Red Hat Enterprise Linux 6 system, read the *Required minimums* section of the [Red Hat Enterprise Linux Technology Capabilities and Limits](#) comparison chart. When the **kdump** crash recovery is enabled, the minimum memory requirements increase by the amount of memory reserved for it. This value is determined by the user, and defaults to 128 MB plus 64 MB for each TB of physical memory (that is, a total of 192 MB for a system with 1 TB of physical memory). The memory can be attempted up to the maximum of 896 MB if required. This is recommended especially in large environments, for example in systems with a large number of Logical Unit Numbers (LUNs).

## The Target Settings Tab

The **Target Settings** tab enables you to specify the target location for the **vmcore** dump. It can be either stored as a file in a local file system, written directly to a device, or sent over a network using the NFS (Network File System) or SSH (Secure Shell) protocol.





**Figure 32.2. Target Settings**

To save the dump to the local file system, select the **Local filesystem** radio button. Optionally, you can customize the settings by choosing a different partition from the **Partition**, and a target directory from the **Path** pulldown lists.

To write the dump directly to a device, select the **Raw device** radio button, and choose the desired target device from the pulldown list next to it.

To store the dump to a remote machine, select the **Network** radio button. To use the NFS protocol, select the **NFS** radio button, and fill the **Server name** and **Path to directory** fields. To use the SSH protocol, select the **SSH** radio button, and fill the **Server name**, **Path to directory**, and **User name** fields with the remote server address, target directory, and a valid remote user name respectively. See [Chapter 14, OpenSSH](#) for information on how to configure an SSH server, and how to set up a key-based authentication.

**NOTE**

When using Direct-Access Storage Devices (DASDs) as the kdump target, the devices must be specified in the `/etc/dasd.conf` file with other DASDs, for example:

```
0.0.2098
0.0.2198
0.0.2298
0.0.2398
```

Where `0.0.2298` and `0.0.2398` are the DASDs used as the kdump target.

Similarly, when using FCP-attached Small Computer System Interface (SCSI) disks as the kdump target, the disks must be specified in the `/etc/zfcp.conf` file with other FCP-Attached SCSI disks, for example:

```
0.0.3d0c 0x500507630508c1ae 0x402424aa00000000
0.0.3d0c 0x500507630508c1ae 0x402424ab00000000
0.0.3d0c 0x500507630508c1ae 0x402424ac00000000
```

Where `0.0.3d0c 0x500507630508c1ae 0x402424ab00000000` and `0.0.3d0c 0x500507630508c1ae 0x402424ac00000000` are the FCP-attached SCSI disks used as the kdump target.

See the [Adding DASDs](#) and [Adding FCP-Attached Logical Units \(LUNs\)](#) chapters in the [Installation Guide](#) for Red Hat Enterprise Linux 6 for detailed information about configuring DASDs and FCP-attached SCSI disks.

**IMPORTANT**

When transferring a core file to a remote target over SSH, the core file needs to be serialized for the transfer. This creates a `vmcore.flat` file in the `/var/crash/` directory on the target system, which is unreadable by the `crash` utility. To convert `vmcore.flat` to a dump file that is readable by `crash`, run the following command as root on the target system:

```
~]# /usr/sbin/makedumpfile -R */tmp/vmcore-rearranged* <
*vmcore.flat*
```

For a complete list of currently supported targets, see [Table 32.1, “Supported kdump targets”](#).

**Table 32.1. Supported kdump targets**

Type	Supported Targets	Unsupported Targets
Raw device	All locally attached raw disks and partitions.	—
Local file system	<b>ext2</b> , <b>ext3</b> , <b>ext4</b> , <b>minix</b> , <b>btrfs</b> and <b>xfs</b> file systems on directly attached disk drives, hardware RAID logical drives, LVM devices, and <b>mdraid</b> arrays.	Any local file system not explicitly listed as supported in this table, including the <b>auto</b> type (automatic file system detection).

Type	Supported Targets	Unsupported Targets
Remote directory	Remote directories accessed using the <b>NFS</b> or <b>SSH</b> protocol over <b>IPv4</b> .	Remote directories on the <b>rootfs</b> file system accessed using the <b>NFS</b> protocol.
	Remote directories accessed using the <b>iSCSI</b> protocol over software initiators, unless <b>iBFT</b> ( <i>iSCSI Boot Firmware Table</i> ) is utilized.	Remote directories accessed using the <b>iSCSI</b> protocol using <b>iBFT</b> .
	Multipath-based storages.[a]	Remote directories accessed using the <b>iSCSI</b> protocol over hardware initiators.
	—	Remote directories accessed over <b>IPv6</b> .
	—	Remote directories accessed using the <b>SMB/CIFS</b> protocol.
—	Remote directories accessed using the <b>FCoE</b> ( <i>Fibre Channel over Ethernet</i> ) protocol.	
—	Remote directories accessed using wireless network interfaces.	
[a] Supported in Red Hat Enterprise Linux 6 from kexec-tools-2.0.0-245.el6 onwards.		

### The Filtering Settings Tab

The **Filtering Settings** tab enables you to select the filtering level for the **vmcore** dump.

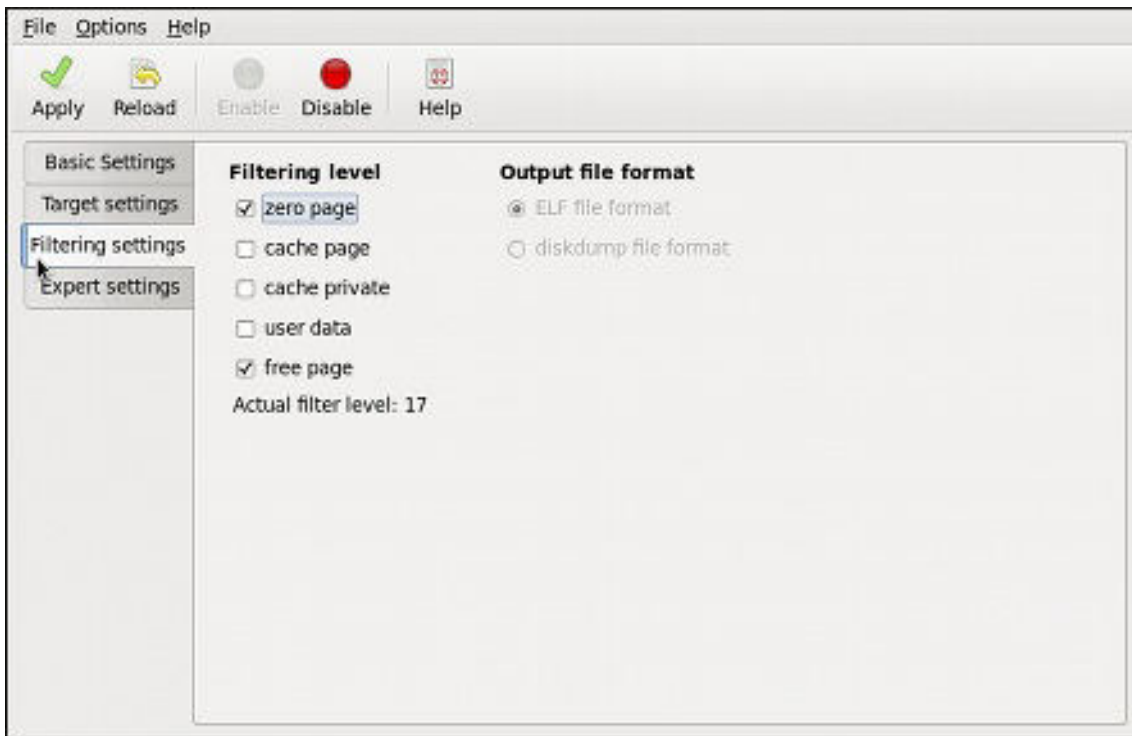


Figure 32.3. Filtering Settings

To exclude the **zero page**, **cache page**, **cache private**, **user data**, or **free page** from the dump, select the check box next to the appropriate label.

### The Expert Settings Tab

The **Expert Settings** tab enables you to choose which kernel and initial RAM disk to use, as well as to customize the options that are passed to the kernel and the core collector program.

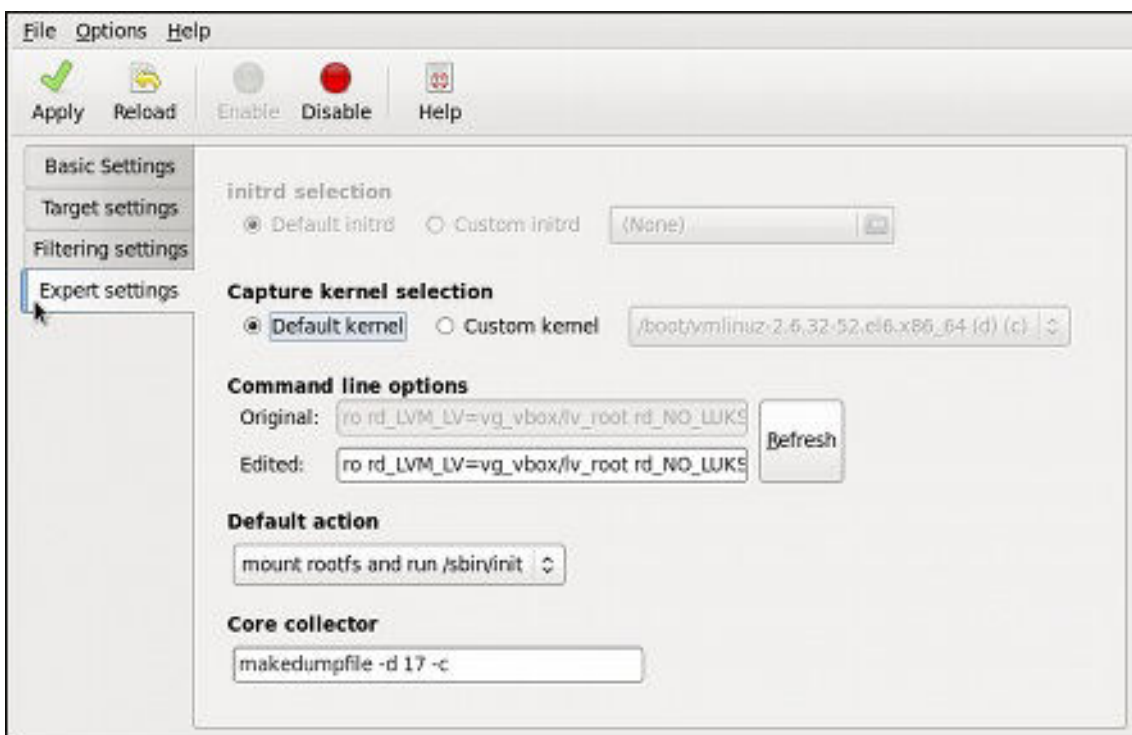


Figure 32.4. Expert Settings

To use a different initial RAM disk, select the **Custom initrd** radio button, and choose the desired RAM disk from the pulldown list next to it.

To capture a different kernel, select the **Custom kernel** radio button, and choose the desired kernel image from the pulldown list on the right.

To adjust the list of options that are passed to the kernel at boot time, edit the content of the **Edited** text field. Note that you can always revert your changes by clicking the **Refresh** button.

To choose what action to perform when **kdump** fails to create a core dump, select an appropriate option from the **Default action** pulldown list. Available options are **mount rootfs and run /sbin/init** (the default action), **reboot** (to reboot the system), **shell** (to present a user with an interactive shell prompt), **halt** (to halt the system), and **poweroff** (to power the system off).

To customize the options that are passed to the **makedumpfile** core collector, edit the **Core collector** text field; see [the section called “Configuring the Core Collector”](#) for more information.

### 32.2.3. Configuring kdump on the Command Line

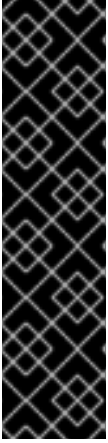
#### Configuring the Memory Usage

Memory reserved for the kdump kernel is always reserved during system boot, which means that the amount of memory is specified in the system's boot loader configuration. This section will explain how to change the amount of reserved memory on AMD64 and Intel 64 systems and IBM Power Systems servers using the **GRUB** boot loader, and on IBM System z using **zipl**. To configure the amount of memory to be reserved for the **kdump** kernel, edit the `/boot/grub/grub.conf` file and add **crashkernel=<size>M** or **crashkernel=auto** to the list of kernel options as shown in [Example 32.1, “A sample /boot/grub/grub.conf file”](#). Note that the **crashkernel=auto** option only reserves the memory if the physical memory of the system is equal to or greater than:

- **2 GB** on 32-bit and 64-bit **x86** architectures;
- **2 GB** on **PowerPC** if the page size is 4 KB, or **8 GB** otherwise;
- **4 GB** on **IBM S/390**.

#### Example 32.1. A sample /boot/grub/grub.conf file

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this
file
# NOTICE: You have a /boot partition. This means that
#           all kernel and initrd paths are relative to /boot/, eg.
#           root (hd0,0)
#           kernel /vmlinuz-version ro root=/dev/sda3
#           initrd /initrd
#boot=/dev/sda
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (2.6.32-220.el6.x86_64)
    root (hd0,0)
    kernel /vmlinuz-2.6.32-220.el6.x86_64 ro root=/dev/sda3
crashkernel=128M
    initrd /initramfs-2.6.32-220.el6.x86_64.img
```



## IMPORTANT

This section is available only if the system has enough memory. To learn about minimum memory requirements of the Red Hat Enterprise Linux 6 system, read the *Required minimums* section of the [Red Hat Enterprise Linux Technology Capabilities and Limits](#) comparison chart. When the **kdump** crash recovery is enabled, the minimum memory requirements increase by the amount of memory reserved for it. This value is determined by the user, and defaults to 128 MB plus 64 MB for each TB of physical memory (that is, a total of 192 MB for a system with 1 TB of physical memory). The memory can be attempted up to the maximum of 896 MB if required. This is recommended especially in large environments, for example in systems with a large number of Logical Unit Numbers (LUNs).

### Configuring the Target Type

When a kernel crash is captured, the core dump can be either stored as a file in a local file system, written directly to a device, or sent over a network using the NFS (Network File System) or SSH (Secure Shell) protocol. Only one of these options can be set at the moment, and the default option is to store the **vmcore** file in the **/var/crash/** directory of the local file system. To change this, as **root**, open the **/etc/kdump.conf** configuration file in a text editor and edit the options as described below.

To change the local directory in which the core dump is to be saved, remove the hash sign (“#”) from the beginning of the **#path /var/crash** line, and replace the value with a desired directory path. Optionally, if you want to write the file to a different partition, follow the same procedure with the **#ext4 /dev/sda3** line as well, and change both the file system type and the device (a device name, a file system label, and UUID are all supported) accordingly. For example:

```
ext3 /dev/sda4
path /usr/local/cores
```

To write the dump directly to a device, remove the hash sign (“#”) from the beginning of the **#raw /dev/sda5** line, and replace the value with a desired device name. For example:

```
raw /dev/sdb1
```

To store the dump to a remote machine using the NFS protocol, remove the hash sign (“#”) from the beginning of the **#net my.server.com:/export/tmp** line, and replace the value with a valid host name and directory path. For example:

```
net penguin.example.com:/export/cores
```

To store the dump to a remote machine using the SSH protocol, remove the hash sign (“#”) from the beginning of the **#net user@my.server.com** line, and replace the value with a valid user name and host name. For example:

```
net john@penguin.example.com
```

See [Chapter 14, OpenSSH](#) for information on how to configure an SSH server, and how to set up a key-based authentication.

For a complete list of currently supported targets, see [Table 32.1, “Supported kdump targets”](#).

**NOTE**

When using Direct-Access Storage Devices (DASDs) as the `kdump` target, the devices must be specified in the `/etc/dasd.conf` file with other DASDs, for example:

```
0.0.2098
0.0.2198
0.0.2298
0.0.2398
```

Where `0.0.2298` and `0.0.2398` are the DASDs used as the `kdump` target.

Similarly, when using FCP-attached Small Computer System Interface (SCSI) disks as the `kdump` target, the disks must be specified in the `/etc/zfcp.conf` file with other FCP-Attached SCSI disks, for example:

```
0.0.3d0c 0x500507630508c1ae 0x402424aa00000000
0.0.3d0c 0x500507630508c1ae 0x402424ab00000000
0.0.3d0c 0x500507630508c1ae 0x402424ac00000000
```

Where `0.0.3d0c 0x500507630508c1ae 0x402424ab00000000` and `0.0.3d0c 0x500507630508c1ae 0x402424ac00000000` are the FCP-attached SCSI disks used as the `kdump` target.

See the [Adding DASDs](#) and [Adding FCP-Attached Logical Units \(LUNs\)](#) chapters in the [Installation Guide](#) for Red Hat Enterprise Linux 6 for detailed information about configuring DASDs and FCP-attached SCSI disks.

**IMPORTANT**

When transferring a core file to a remote target over SSH, the core file needs to be serialized for the transfer. This creates a `vmcore.flat` file in the `/var/crash/` directory on the target system, which is unreadable by the `crash` utility. To convert `vmcore.flat` to a dump file that is readable by `crash`, run the following command as `root` on the target system:

```
~]# /usr/sbin/makedumpfile -R */tmp/vmcore-rearranged* <
*vmcore.flat*
```

**Configuring the Core Collector**

To reduce the size of the `vmcore` dump file, `kdump` allows you to specify an external application (that is, a core collector) to compress the data, and optionally leave out all irrelevant information. Currently, the only fully supported core collector is `makedumpfile`.

To enable the core collector, as `root`, open the `/etc/kdump.conf` configuration file in a text editor, remove the hash sign (“#”) from the beginning of the `#core_collector makedumpfile -c --message-level 1 -d 31` line, and edit the command-line options as described below.

To enable the dump file compression, add the `-c` parameter. For example:

```
core_collector makedumpfile -c
```

To remove certain pages from the dump, add the **-d** *value* parameter, where *value* is a sum of values of pages you want to omit as described in [Table 32.2, “Supported filtering levels”](#). For example, to remove both zero and free pages, use the following:

```
core_collector makedumpfile -d 17 -c
```

See the manual page for **makedumpfile** for a complete list of available options.

**Table 32.2. Supported filtering levels**

Option	Description
<b>1</b>	Zero pages
<b>2</b>	Cache pages
<b>4</b>	Cache private
<b>8</b>	User pages
<b>16</b>	Free pages

### Changing the Default Action

By default, when **kdump** fails to create a core dump, the root file system is mounted and **/sbin/init** is run. To change this behavior, as **root**, open the **/etc/kdump.conf** configuration file in a text editor, remove the hash sign (“#”) from the beginning of the **#default shell** line, and replace the value with a desired action as described in [Table 32.3, “Supported actions”](#).

**Table 32.3. Supported actions**

Option	Description
<b>reboot</b>	Reboot the system, losing the core in the process.
<b>halt</b>	Halt the system.
<b>poweroff</b>	Power off the system.
<b>shell</b>	Run the <b>msh</b> session from within the initramfs, allowing a user to record the core manually.

For example:

```
default halt
```

### Enabling the Service

To start the **kdump** daemon at boot time, type the following at a shell prompt as **root**:

```
chkconfig kdump on
```



This will enable the service for runlevels **2**, **3**, **4**, and **5**. Similarly, typing **chkconfig kdump off** will disable it for all runlevels. To start the service in the current session, use the following command as **root**:

```
service kdump start
```

For more information on runlevels and configuring services in general, see [Chapter 12, Services and Daemons](#).

### 32.2.4. Testing the Configuration



#### WARNING

The commands below will cause the kernel to crash. Use caution when following these steps, and by no means use them on a production machine.

To test the configuration, reboot the system with **kdump** enabled, and make sure that the service is running (see [Section 12.3, “Running Services”](#) for more information on how to run a service in Red Hat Enterprise Linux):

```
~]# service kdump status  
Kdump is operational
```

Then type the following commands at a shell prompt:

```
echo 1 > /proc/sys/kernel/sysrq  
echo c > /proc/sysrq-trigger
```

This will force the Linux kernel to crash, and the **address-YYYY-MM-DD-HH:MM:SS/vmcore** file will be copied to the location you have selected in the configuration (that is, to **/var/crash/** by default).

## 32.3. ANALYZING THE CORE DUMP

To determine the cause of the system crash, you can use the **crash** utility, which provides an interactive prompt very similar to the GNU Debugger (GDB). This utility allows you to interactively analyze a running Linux system as well as a core dump created by **netdump**, **diskdump**, **xendump**, or **kdump**.

**IMPORTANT**

To analyze the **vmcore** dump file, you must have the **crash** and **kernel-debuginfo** packages installed. To install the **crash** package in your system, type the following at a shell prompt as **root**:

```
yum install crash
```

To install the **kernel-debuginfo** package, make sure that you have the **yum-utils** package installed and run the following command as **root**:

```
debuginfo-install kernel
```

Note that in order to use this command, you need to have access to the repository with debugging packages. If your system is registered with Red Hat Subscription Management, enable the **rhel-6-variant-debug-rpms** repository as described in [Section 8.4.4, “Viewing the Current Configuration”](#). If your system is registered with RHN Classic, subscribe the system to the **rhel-architecture-variant-6-debuginfo** channel as documented here: <https://access.redhat.com/site/solutions/9907>.

**32.3.1. Running the crash Utility**

To start the utility, type the command in the following form at a shell prompt:

```
crash /usr/lib/debug/lib/modules/kernel/vmlinux  
/var/crash/timestamp/vmcore
```

Note that the *kernel* version should be the same that was captured by **kdump**. To find out which kernel you are currently running, use the **uname -r** command.

**Example 32.2. Running the crash utility**

```
~]# crash /usr/lib/debug/lib/modules/2.6.32-69.el6.i686/vmlinux \  
/var/crash/127.0.0.1-2010-08-25-08:45:02/vmcore  
  
crash 5.0.0-23.el6  
Copyright (C) 2002-2010 Red Hat, Inc.  
Copyright (C) 2004, 2005, 2006 IBM Corporation  
Copyright (C) 1999-2006 Hewlett-Packard Co  
Copyright (C) 2005, 2006 Fujitsu Limited  
Copyright (C) 2006, 2007 VA Linux Systems Japan K.K.  
Copyright (C) 2005 NEC Corporation  
Copyright (C) 1999, 2002, 2007 Silicon Graphics, Inc.  
Copyright (C) 1999, 2000, 2001, 2002 Mission Critical Linux, Inc.  
This program is free software, covered by the GNU General Public  
License,  
and you are welcome to change it and/or distribute copies of it under  
certain conditions. Enter "help copying" to see the conditions.  
This program has absolutely no warranty. Enter "help warranty" for  
details.  
  
GNU gdb (GDB) 7.0  
Copyright (C) 2009 Free Software Foundation, Inc.  
License GPLv3+: GNU GPL version 3 or later
```

```

<http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show
copying"
and "show warranty" for details.
This GDB was configured as "i686-pc-linux-gnu"...

        KERNEL: /usr/lib/debug/lib/modules/2.6.32-69.el6.i686/vmlinux
        DUMPFILE: /var/crash/127.0.0.1-2010-08-25-08:45:02/vmcore [PARTIAL
DUMP]
        CPUS: 4
        DATE: Wed Aug 25 08:44:47 2010
        UPTIME: 00:09:02
LOAD AVERAGE: 0.00, 0.01, 0.00
        TASKS: 140
        NODENAME: hp-dl320g5-02.lab.bos.redhat.com
        RELEASE: 2.6.32-69.el6.i686
        VERSION: #1 SMP Tue Aug 24 10:31:45 EDT 2010
        MACHINE: i686 (2394 Mhz)
        MEMORY: 8 GB
        PANIC: "Oops: 0002 [#1] SMP " (check log for details)
        PID: 5591
        COMMAND: "bash"
        TASK: f196d560 [THREAD_INFO: ef4da000]
        CPU: 2
        STATE: TASK_RUNNING (PANIC)

crash>

```

### 32.3.2. Displaying the Message Buffer

To display the kernel message buffer, type the **log** command at the interactive prompt.

#### Example 32.3. Displaying the kernel message buffer

```

crash> log
... several lines omitted ...
EIP: 0060:[<c068124f>] EFLAGS: 00010096 CPU: 2
EIP is at sysrq_handle_crash+0xf/0x20
EAX: 00000063 EBX: 00000063 ECX: c09e1c8c EDX: 00000000
ESI: c0a09ca0 EDI: 00000286 EBP: 00000000 ESP: ef4dbf24
DS: 007b ES: 007b FS: 00d8 GS: 00e0 SS: 0068
Process bash (pid: 5591, ti=ef4da000 task=f196d560 task.ti=ef4da000)
Stack:
c068146b c0960891 c0968653 00000003 00000000 00000002 efade5c0 c06814d0
<0> ffffffff c068150f b7776000 f2600c40 c0569ec4 ef4dbf9c 00000002
b7776000
<0> efade5c0 00000002 b7776000 c0569e60 c051de50 ef4dbf9c f196d560
ef4dbfb4
Call Trace:
[<c068146b>] ? __handle_sysrq+0xfb/0x160
[<c06814d0>] ? write_sysrq_trigger+0x0/0x50
[<c068150f>] ? write_sysrq_trigger+0x3f/0x50
[<c0569ec4>] ? proc_reg_write+0x64/0xa0

```

```

[<c0569e60>] ? proc_reg_write+0x0/0xa0
[<c051de50>] ? vfs_write+0xa0/0x190
[<c051e8d1>] ? sys_write+0x41/0x70
[<c0409adc>] ? syscall_call+0x7/0xb
Code: a0 c0 01 0f b6 41 03 19 d2 f7 d2 83 e2 03 83 e0 cf c1 e2 04 09 d0
88 41 03 f3 c3 90 c7 05 c8 1b 9e c0 01 00 00 00 0f ae f8 89 f6 <c6> 05
00 00 00 00 01 c3 89 f6 8d bc 27 00 00 00 00 8d 50 d0 83
EIP: [<c068124f>] sysrq_handle_crash+0xf/0x20 SS:ESP 0068:ef4dbf24
CR2: 0000000000000000

```

Type **help log** for more information on the command usage.



## NOTE

The kernel message buffer includes the most essential information about the system crash and, as such, it is always dumped first in to the **vmcore-dmesg.txt** file. This is useful when an attempt to get the full **vmcore** file failed, for example because of lack of space on the target location. By default, **vmcore-dmesg.txt** is located in the **/var/crash/** directory.

### 32.3.3. Displaying a Backtrace

To display the kernel stack trace, type the **bt** command at the interactive prompt. You can use **bt pid** to display the backtrace of the selected process.

#### Example 32.4. Displaying the kernel stack trace

```

crash> bt
PID: 5591  TASK: f196d560  CPU: 2  COMMAND: "bash"
#0 [ef4dbdcc] crash_kexec at c0494922
#1 [ef4dbe20] oops_end at c080e402
#2 [ef4dbe34] no_context at c043089d
#3 [ef4dbe58] bad_area at c0430b26
#4 [ef4dbe6c] do_page_fault at c080fb9b
#5 [ef4dbee4] error_code (via page_fault) at c080d809
   EAX: 00000063  EBX: 00000063  ECX: c09e1c8c  EDX: 00000000  EBP:
00000000
   DS: 007b      ESI: c0a09ca0  ES: 007b      EDI: 00000286  GS:
00e0
   CS: 0060      EIP: c068124f  ERR: ffffffff  EFLAGS: 00010096
#6 [ef4dbf18] sysrq_handle_crash at c068124f
#7 [ef4dbf24] __handle_sysrq at c0681469
#8 [ef4dbf48] write_sysrq_trigger at c068150a
#9 [ef4dbf54] proc_reg_write at c0569ec2
#10 [ef4dbf74] vfs_write at c051de4e
#11 [ef4dbf94] sys_write at c051e8cc
#12 [ef4dbfb0] system_call at c0409ad5
   EAX: ffffffff  EBX: 00000001  ECX: b7776000  EDX: 00000002
   DS: 007b      ESI: 00000002  ES: 007b      EDI: b7776000
   SS: 007b      ESP: bfc2088  EBP: bfc20b4  GS: 0033
   CS: 0073      EIP: 00edc416  ERR: 00000004  EFLAGS: 00000246

```

Type `help bt` for more information on the command usage.

### 32.3.4. Displaying a Process Status

To display status of processes in the system, type the `ps` command at the interactive prompt. You can use `ps pid` to display the status of the selected process.

#### Example 32.5. Displaying status of processes in the system

```
crash> ps
  PID   PPID  CPU  TASK      ST  %MEM   VSZ   RSS  COMM
>    0     0   0  c09dc560  RU   0.0     0     0  [swapper]
>    0     0   1  f7072030  RU   0.0     0     0  [swapper]
    0     0   2  f70a3a90  RU   0.0     0     0  [swapper]
>    0     0   3  f70ac560  RU   0.0     0     0  [swapper]
    1     0   1  f705ba90  IN   0.0  2828  1424  init
... several lines omitted ...
 5566     1   1  f2592560  IN   0.0  12876   784  auditd
 5567     1   2  ef427560  IN   0.0  12876   784  auditd
 5587   5132   0  f196d030  IN   0.0  11064  3184  sshd
> 5591   5587   2  f196d560  RU   0.0   5084  1648  bash
```

Type `help ps` for more information on the command usage.

### 32.3.5. Displaying Virtual Memory Information

To display basic virtual memory information, type the `vm` command at the interactive prompt. You can use `vm pid` to display information on the selected process.

#### Example 32.6. Displaying virtual memory information of the current context

```
crash> vm
PID: 5591   TASK: f196d560  CPU: 2   COMMAND: "bash"
  MM      PGD      RSS      TOTAL_VM
f19b5900  ef9c6000  1648k    5084k
  VMA      START      END      FLAGS  FILE
f1bb0310  242000    260000  8000875  /lib/ld-2.12.so
f26af0b8  260000    261000  8100871  /lib/ld-2.12.so
efbc275c  261000    262000  8100873  /lib/ld-2.12.so
efbc2a18  268000    3ed000  8000075  /lib/libc-2.12.so
efbc23d8  3ed000    3ee000  8000070  /lib/libc-2.12.so
efbc2888  3ee000    3f0000  8100071  /lib/libc-2.12.so
efbc2cd4  3f0000    3f1000  8100073  /lib/libc-2.12.so
efbc243c  3f1000    3f4000  100073
efbc28ec  3f6000    3f9000  8000075  /lib/libdl-2.12.so
efbc2568  3f9000    3fa000  8100071  /lib/libdl-2.12.so
efbc2f2c  3fa000    3fb000  8100073  /lib/libdl-2.12.so
f26af888  7e6000    7fc000  8000075  /lib/libtinfo.so.5.7
f26aff2c  7fc000    7ff000  8100073  /lib/libtinfo.so.5.7
efbc211c  d83000    d8f000  8000075  /lib/libnss_files-2.12.so
efbc2504  d8f000    d90000  8100071  /lib/libnss_files-2.12.so
efbc2950  d90000    d91000  8100073  /lib/libnss_files-2.12.so
f26afe00  edc000    edd000  4040075
```

```
f1bb0a18 8047000 8118000 8001875 /bin/bash
f1bb01e4 8118000 811d000 8101873 /bin/bash
f1bb0c70 811d000 8122000 100073
f26afae0 9fd9000 9ffa000 100073
... several lines omitted ...
```

Type `help vm` for more information on the command usage.

### 32.3.6. Displaying Open Files

To display information about open files, type the `files` command at the interactive prompt. You can use `files pid` to display files opened by the selected process.

#### Example 32.7. Displaying information about open files of the current context

```
crash> files
PID: 5591  TASK: f196d560  CPU: 2  COMMAND: "bash"
ROOT: /  CWD: /root
FD  FILE  DENTRY  INODE  TYPE  PATH
0  f734f640  eedc2c6c  eecd6048  CHR  /pts/0
1  efade5c0  eee14090  f00431d4  REG  /proc/sysrq-trigger
2  f734f640  eedc2c6c  eecd6048  CHR  /pts/0
10  f734f640  eedc2c6c  eecd6048  CHR  /pts/0
255  f734f640  eedc2c6c  eecd6048  CHR  /pts/0
```

Type `help files` for more information on the command usage.

### 32.3.7. Exiting the Utility

To exit the interactive prompt and terminate `crash`, type `exit` or `q`.

#### Example 32.8. Exiting the crash utility

```
crash> exit
~]#
```

## 32.4. USING FADUMP ON IBM POWERPC HARDWARE

Starting with Red Hat Enterprise Linux 6.8 an alternative dumping mechanism to `kdump`, the **firmware-assisted dump** (`fadump`), is available. The `fadump` feature is supported only on IBM Power Systems. The goal of `fadump` is to enable the dump of a crashed system, and to do so from a fully-reset system, and to minimize the total elapsed time until the system is back in production use. The `fadump` feature is integrated with `kdump` infrastructure present in the user space to seamlessly switch between `kdump` and `fadump` mechanisms.

Firmware-assisted dump (`fadump`) is a reliable alternative to `kexec-kdump` available on IBM PowerPC LPARS. It captures vmcore from a fully-reset system with PCI and I/O devices reinitialized. While this mechanism uses the firmware to preserve the memory in case of a crash, it reuses the `kdump` userspace

scripts to save the vmcore"

To achieve this, **fadump** registers the regions of memory that must be preserved in the event of a crash with the system firmware. These regions consist of all the system memory contents, except the boot memory, system registers and hardware Page Table Entries (PTEs).



## NOTE

The area of memory not preserved and known as **boot memory** is the amount of RAM required to successfully boot the kernel after a crash event. By default, the boot memory size is 256MB or 5% of total system RAM, whichever is larger.

Unlike a **kexec**-initiated event, the **fadump** process uses the production kernel to recover a crash dump. When booting after a crash, PowerPC hardware makes the device node **/proc/device-tree/rtas/ibm, kernel-dump** available to **procf**s, which the fadump-aware **kdump** scripts check for to save the vmcore. After this has completed, the system is rebooted cleanly.

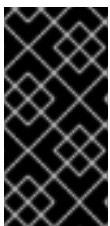
## Enabling fadump

1. Install and configure **kdump** as described in [Section 32.1, “Installing the kdump Service”](#) and [Section 32.2, “Configuring the kdump Service”](#).
2. Add **fadump=on** to the **GRUB\_CMDLINE\_LINUX** line in **/etc/default/grub**:

```
GRUB_CMDLINE_LINUX="rd.lvm.lv=rhel/swap crashkernel=auto
rd.lvm.lv=rhel/root rhgb quiet fadump=on"
```

3. (optional) If you want to specify reserved boot memory instead of accepting the defaults, add **fadump\_reserve\_mem=xxM** to **GRUB\_CMDLINE\_LINUX** in **/etc/default/grub**, where **xx** is the amount of the memory required in megabytes:

```
GRUB_CMDLINE_LINUX="rd.lvm.lv=rhel/swap crashkernel=auto
rd.lvm.lv=rhel/root rhgb quiet fadump=on fadump_reserve_mem=xxM"
```



## IMPORTANT

As with all boot configuration options, it is strongly recommended that you test the configuration before it is needed. If you observe Out of Memory (OOM) errors when booting from the crash kernel, increase the value specified in **fadump\_reserve\_mem=** until the crash kernel can boot cleanly. Some trial and error may be required in this case.

## 32.5. ADDITIONAL RESOURCES

### Installed Documentation

- **kdump.conf(5)** — a manual page for the **/etc/kdump.conf** configuration file containing the full documentation of available options.
- **makedumpfile(8)** — a manual page for the **makedumpfile** core collector.
- **kexec(8)** — a manual page for **kexec**.
- **crash(8)** — a manual page for the **crash** utility.

- `/usr/share/doc/kexec-tools-version/kexec-kdump-howto.txt` — an overview of the **kdump** and **kexec** installation and usage.
- `/usr/share/doc/kexec-tools-version/fadump-howto.txt` — more details about the **fadump** mechanism, including PowerPC-specific methods of resetting hardware. Replace *version* with kexec-tools version installed on your system.

## Useful Websites

<https://access.redhat.com/kb/docs/DOC-6039>

The Red Hat Knowledgebase article about the **kexec** and **kdump** configuration.

<https://access.redhat.com/kb/docs/DOC-45183>

The Red Hat Knowledgebase article about supported **kdump** targets.

<http://people.redhat.com/anderson/>

The **crash** utility homepage.



## **PART IX. SYSTEM RECOVERY**

This part introduces system recovery modes on Red Hat Enterprise Linux 6 and advises users on how to repair the system in certain situations. It also describes how to use the Relax-and-Recover (ReaR) disaster recovery and system migration utility.

## CHAPTER 33. SYSTEM RECOVERY

Red Hat Enterprise Linux 6 offers three system recovery modes, **rescue mode**, **single-user mode**, and **emergency mode** that can be used to repair malfunctioning systems. This chapter describes how to boot into each system recovery mode and gives guidance to resolve certain problems that can only be solved with help of system recovery modes.

These are the usual reasons why you may need to boot to one of the system recovery modes:

- You are unable to boot normally into Red Hat Enterprise Linux (runlevel 3 or 5).
- You need to resolve hardware or software problems that cannot be resolved while the system is running normally, or you want to access some important files off of your hard drive.
- You forgot the root password.

Some of the problems behind are further discussed in [Section 33.4, “Resolving Problems in System Recovery Modes”](#).

### 33.1. RESCUE MODE

**Rescue mode** provides the ability to boot a small Red Hat Enterprise Linux environment entirely from external media, such as CD-ROM or USB drive, instead of the system’s hard drive. It contains command-line utilities for repairing a wide variety of issues. In this mode, you can mount file systems as read-only or even to not mount them at all, blacklist or add drivers provided on a driver disc, install or upgrade system packages, or manage partitions.

To boot into **rescue mode** follow this procedure:

#### Procedure 33.1. Booting into Rescue Mode

1. Boot the system from either minimal boot media, or a full installation DVD or USB drive, and wait for the boot menu to appear. For details about booting the system from the chosen media, see the respective chapters in the [Installation Guide](#).
2. From the boot menu, append the **rescue** keyword as a kernel parameter to the boot command line.
3. If your system requires a third-party driver provided on a *driver disc* to boot, append the additional option **dd** to the boot command line to load that driver:

```
rescue dd
```

For more information about using a disc driver at boot time, see the respective chapters in the [Installation Guide](#).

4. If a driver that is a part of the Red Hat Enterprise Linux 6 distribution prevents the system from booting, blacklist that driver by appending the **rdblacklist** option to the boot command line:

```
rescue rdblacklist=driver_name
```

5. Answer a few basic questions and select the location of a valid rescue image as you are prompted to. Select the relevant type from **Local CD-ROM**, **Hard Drive**, **NFS image**, **FTP**, or **HTTP**. The selected location must contain a valid installation tree, and the installation tree must

be for the same version of Red Hat Enterprise Linux as is the disk from which you booted. For more information about how to setup an installation tree on a hard drive, NFS server, FTP server, or HTTP server, see the respective chapters in the [Installation Guide](#).

If you select a rescue image that does not require a network connection, you are asked whether or not you want to establish a network connection. A network connection is useful if you need to backup files to a different computer or install some RPM packages from a shared network location.

6. The following message is displayed:

```
The rescue environment will now attempt to find your Linux
installation and mount it under the directory /mnt/sysimage. You can
then make any changes required to your system. If you want to
proceed with this step choose 'Continue'. You can also choose to
mount your file systems read-only instead of read-write by choosing
'Read-only'. If for some reason this process fails you can choose
'Skip' and this step will be skipped and you will go directly to a
command shell.
```

If you select **Continue**, the system attempts to mount your root partition under the `/mnt/sysimage/` directory. The root partition typically contains several file systems, such as `/home/`, `/boot/`, and `/var/`, which are automatically mounted to the correct locations. If mounting the partition fails, you will be notified. If you select **Read-Only**, the system attempts to mount your file systems under the directory `/mnt/sysimage/`, but in read-only mode. If you select **Skip**, your file systems will not be mounted. Choose **Skip** if you think your file system is corrupted.

7. Once you have your system in rescue mode, the following prompt appears on the virtual console (VC) 1 and VC 2. Use the **Ctrl-Alt-F1** key combination to access VC 1 and **Ctrl-Alt-F2** to access VC 2:

```
sh-3.00b#
```

If you selected **Continue** to mount your partitions automatically and they were mounted successfully, you are in **single-user mode**.

Even if your file system is mounted, the default root partition while in **rescue mode** is a temporary root partition, not the root partition of the file system used during normal user mode (runlevel 3 or 5). If you selected to mount your file system and it mounted successfully, you can change the root partition of the rescue mode environment to the root partition of your file system by executing the following command:

```
sh-3.00b# chroot /mnt/sysimage
```

This is useful if you need to run commands, such as **rpm**, that require your root partition to be mounted as `/`. To exit the **chroot** environment, type **exit** to return to the prompt.

If you selected **Skip**, you can still try to mount a partition or a LVM2 logical volume manually inside **rescue mode** by creating a directory and typing the following command:

```
sh-3.00b# mkdir /directory
sh-3.00b# mount -t ext4 /dev/mapper/VolGroup00-LogVol02 /directory
```

where */directory* is a directory that you have created and */dev/mapper/VolGroup00-LogVol02* is the LVM2 logical volume you want to mount. If the partition is of **ext2** or **ext3** type, replace **ext4** with **ext2** or **ext3** respectively.

If you do not know the names of all physical partitions, use the following command to list them:

```
sh-3.00b# fdisk -l
```

If you do not know the names of all LVM2 physical volumes, volume groups, or logical volumes, use the **pvdisplay**, **vgdisplay** or **lvdisplay** commands, respectively.

From the prompt, you can run many useful commands, such as:

- **ssh**, **scp**, and **ping** if the network is started
- **dump** and **restore** for users with tape drives
- **parted** and **fdisk** for managing partitions
- **rpm** for installing or upgrading software
- **vi** for editing text files

## 33.2. SINGLE-USER MODE

**Single-user mode** provides a Linux environment for a single user that allows you to recover your system from problems that cannot be resolved in networked multi-user environment. You do not need an external boot device to be able to boot into **single-user mode**, and you can switch into it directly while the system is running. To switch into **single-user mode** on the running system, issue the following command from the command line:

```
~]# init 1
```

In **single-user mode**, the system boots with your local file systems mounted, many important services running, and a usable maintenance shell that allows you to perform many of the usual system commands. Therefore, **single-user mode** is mostly useful for resolving problems when the system boots but does not function properly or you cannot log into it.



### WARNING

The **single-user mode** automatically tries to mount your local file systems. Booting to **single-user mode** could result in loss of data if any of your local file systems cannot be successfully mounted.

To boot into **single-user mode** follow this procedure:

### Procedure 33.2. Booting into Single-User Mode

1. At the GRUB boot screen, press any key to enter the GRUB interactive menu.
2. Select **Red Hat Enterprise Linux** with the version of the kernel that you want to boot and press the **a** to append the line.
3. Type **single** as a separate word at the end of the line and press **Enter** to exit GRUB edit mode. Alternatively, you can type **1** instead of single.

### 33.3. EMERGENCY MODE

**Emergency mode**, provides the minimal bootable environment and allows you to repair your system even in situations when **rescue mode** is unavailable. In **emergency mode**, the system mounts only the **root** file system, and it is mounted as read-only. Also, the system does not activate any network interfaces and only a minimum of the essential services are set up. The system does not load any **init** scripts, therefore you can still mount file systems to recover data that would be lost during a re-installation if **init** is corrupted or not working.

To boot into **emergency mode** follow this procedure:

#### Procedure 33.3. Booting into Emergency Mode

1. At the GRUB boot screen, press any key to enter the GRUB interactive menu.
2. Select **Red Hat Enterprise Linux** with the version of the kernel that you want to boot and press the **a** to append the line.
3. Type **emergency** as a separate word at the end of the line and press **Enter** to exit GRUB edit mode.

### 33.4. RESOLVING PROBLEMS IN SYSTEM RECOVERY MODES

This section provides several procedures that explain how to resolve some of the most common problems that needs to be addressed in some of the system recovery modes.

The following procedure shows how to reset a **root** password:

#### Procedure 33.4. Resetting a Root Password

1. Boot to **single-user mode** as described in [Procedure 33.2, “Booting into Single-User Mode”](#).
2. Run the **passwd** command from the maintenance shell command line.

One of the most common causes for an unbootable system is overwriting of the Master Boot Record (MBR) that originally contained the GRUB boot loader. If the boot loader is overwritten, you cannot boot Red Hat Enterprise Linux unless you reconfigure the boot loader in **rescue mode**.

To reinstall GRUB on the MBR of your hard drive, proceed with the following procedure:

#### Procedure 33.5. Reinstalling the GRUB Boot Loader

1. Boot to **rescue mode** as described in [Procedure 33.1, “Booting into Rescue Mode”](#). Ensure that you mount the system's root partition in read-write mode.
2. Execute the following command to change the root partition:

```
sh-3.00b# chroot /mnt/sysimage
```

3. Run the following command to reinstall the GRUB boot loader:

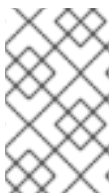
```
sh-3.00b# /sbin/grub-install boot_part
```

where *boot\_part* is your boot partition (typically, **/dev/sda**).

4. Review the **/boot/grub/grub.conf** file, as additional entries may be needed for GRUB to control additional operating systems.
5. Reboot the system.

Another common problem that would render your system unbootable is a change of your root partition number. This can usually happen when resizing a partition or creating a new partition after installation. If the partition number of your root partition changes, the GRUB boot loader might not be able to find it to mount the partition. To fix this problem, boot into **rescue mode** and modify the **/boot/grub/grub.conf** file.

A malfunctioning or missing driver can prevent a system from booting normally. You can use the **RPM** package manager to remove malfunctioning drivers or to add updated or missing drivers in **rescue mode**. If you cannot remove a malfunctioning driver for some reason, you can instead blacklist the driver so that it does not load at boot time.



## NOTE

When you install a driver from a driver disc, the driver disc updates all initramfs images on the system to use this driver. If a problem with a driver prevents a system from booting, you cannot rely on booting the system from another initramfs image.

To remove a malfunctioning driver that prevents the system from booting, follow this procedure:

### Procedure 33.6. Remove a Driver in Rescue Mode

1. Boot to **rescue mode** as described in [Procedure 33.1, “Booting into Rescue Mode”](#). Ensure that you mount the system's root partition in read-write mode.
2. Change the root directory to **/mnt/sysimage/**:

```
sh-3.00b# chroot /mnt/sysimage
```

3. Run the following command to remove the driver package:

```
sh-3.00b# rpm -e driver_name
```

4. Exit the **chroot** environment:

```
sh-3.00b# exit
```

5. Reboot the system.

To install a missing driver that prevents the system from booting, follow this procedure:

### Procedure 33.7. Installing a Driver in Rescue Mode

1. Boot to **rescue mode** as described in [Procedure 33.1, “Booting into Rescue Mode”](#). Ensure that you mount the system's root partition in read-write mode.
2. Mount a media with an RPM package that contains the driver and copy the package to a location of your choice under the `/mnt/sysimage/` directory, for example:  
`/mnt/sysimage/root/drivers/`.

3. Change the root directory to `/mnt/sysimage/`:

```
sh-3.00b# chroot /mnt/sysimage
```

4. Run the following command to install the driver package:

```
sh-3.00b# rpm -ihv /root/drivers/package_name
```

Note that `/root/drivers/` in this chroot environment is `/mnt/sysimage/root/drivers/` in the original rescue environment.

5. Exit the **chroot** environment:

```
sh-3.00b# exit
```

6. Reboot the system.

To blacklist a driver that prevents the system from booting and to ensure that this driver cannot be loaded after the root device is mounted, follow this procedure:

### Procedure 33.8. Blacklisting a Driver in Rescue Mode

1. Boot to **rescue mode** with the command `linux rescue rdblacklist=driver_name`, where `driver_name` is the driver that you need to blacklist. Follow the instructions in [Procedure 33.1, “Booting into Rescue Mode”](#) and ensure that you mount the system's root partition in read-write mode.
2. Open the `/boot/grub/grub.conf` file in the **vi** editor:

```
sh-3.00b# vi /boot/grub/grub.conf
```

3. Identify the default kernel used to boot the system. Each kernel is specified in the `grub.conf` file with a group of lines that begins **title**. The default kernel is specified by the **default** parameter near the start of the file. A value of **0** refers to the kernel described in the first group of lines, a value of **1** refers to the kernel described in the second group, and higher values refer to subsequent kernels in turn.
4. Edit the **kernel1** line of the group to include the option `rdblacklist=driver_name`, where `driver_name` is the driver that you need to blacklist. For example:

```
kernel1 /vmlinuz-2.6.32-71.18-2.el6.i686 ro root=/dev/sda1 rhgb quiet  
rdblacklist=driver_name
```

5. Save the file and exit the **vi** editor by typing:

```
| :wq
```

6. Run the following command to create a new file `/etc/modprobe.d/driver_name.conf` that will ensure blacklisting of the driver after the root partition is mounted:

```
| echo "install driver_name" >  
| /mnt/sysimage/etc/modprobe.d/driver_name.conf
```

7. Reboot the system.



## CHAPTER 34. RELAX-AND-RECOVER (REAR)

When a software or hardware failure breaks the system, the system administrator faces three tasks to restore it to the fully functioning state on a new hardware environment:

1. booting a rescue system on the new hardware
2. replicating the original storage layout
3. restoring user and system files

Most backup software solves only the third problem. To solve the first and second problems, use *Relax-and-Recover (ReaR)*, a disaster recovery and system migration utility.

Backup software creates backups. ReaR complements backup software by creating a *rescue system*. Booting the rescue system on a new hardware allows you to issue the **rear recover** command, which starts the recovery process. During this process, ReaR replicates the partition layout and filesystems, prompts for restoring user and system files from the backup created by backup software, and finally installs the boot loader. By default, the rescue system created by ReaR only restores the storage layout and the boot loader, but not the actual user and system files.

This chapter describes how to use ReaR.

### 34.1. BASIC REAR USAGE

#### 34.1.1. Installing ReaR

Install the rear package by running the following command as root:

```
~]# yum install rear
```

#### 34.1.2. Configuring ReaR

ReaR is configured in the `/etc/rear/local.conf` file. Specify the rescue system configuration by adding these lines:

```
OUTPUT=output format
OUTPUT_URL=output location
```

Substitute *output format* with rescue system format, for example, **ISO** for an ISO disk image or **USB** for a bootable USB.

Substitute *output location* with where it will be put, for example, **file:///mnt/rescue\_system/** for a local filesystem directory or **sftp://backup:password@192.168.0.0/** for an SFTP directory.

##### Example 34.1. Configuring Rescue System Format and Location

To configure ReaR to output the rescue system as an ISO image into the `/mnt/rescue_system/` directory, add these lines to the `/etc/rear/local.conf` file:

```
OUTPUT=ISO
OUTPUT_URL=file:///mnt/rescue_system/
```

See section "Rescue Image Configuration" of the rear(8) man page for a list of all options.

### 34.1.3. Creating a Rescue System

The following example shows how to create a rescue system with verbose output:

```
~]# rear -v mkrescue
Relax-and-Recover 1.17.2 / Git
Using log file: /var/log/rear/rear-rhel68.log
mkdir: created directory `/var/lib/rear/output'
Creating disk layout
Creating root filesystem layout
TIP: To login as root via ssh you need to set up
/root/.ssh/authorized_keys or SSH_ROOT_PASSWORD in your configuration file
Copying files and directories
Copying binaries and libraries
Copying kernel modules
Creating initramfs
Making ISO image
Wrote ISO image: /var/lib/rear/output/rear-rhel68.iso (82M)
Copying resulting files to file location
```

With the configuration from [Example 34.1, “Configuring Rescue System Format and Location”](#), ReaR prints the above output. The last two lines confirm that the rescue system has been successfully created and copied to the configured backup location `/mnt/rescue_system/`. Because the system's host name is `rhel-68`, the backup location now contains directory `rhel-68/` with the rescue system and auxiliary files:

```
~]# ls -lh /mnt/rescue_system/rhel68/
total 82M
-rw-..... 1 root root 202 May 9 11:46 README
-rw-..... 1 root root 160K May 9 11:46 rear.log
-rw-..... 1 root root 82M May 9 11:46 rear-rhel68.iso
-rw-..... 1 root root 275 May 9 11:46 VERSION
```

Transfer the rescue system to an external medium to not lose it in case of a disaster.

### 34.1.4. Scheduling ReaR

To schedule ReaR to regularly create a rescue system using the `cron` job scheduler, add the following line to the `/etc/crontab` file:

```
minute hour day_of_month month day_of_week root /usr/sbin/rear mkrescue
```

Substitute the above command with the cron time specification (described in detail in [Section 27.1.4, “Configuring Cron Jobs”](#)).

#### Example 34.2. Scheduling ReaR

To make ReaR create a rescue system at 22:00 every weekday, add this line to the `/etc/crontab` file:

```
0 22 * * 1-5 root /usr/sbin/rear mkrescue
```



### 34.1.5. Performing a System Rescue

To perform a restore or migration:

1. Boot the rescue system on the new hardware. For example, burn the ISO image to a DVD and boot from the DVD.
2. In the console interface, select the "Recover" option:

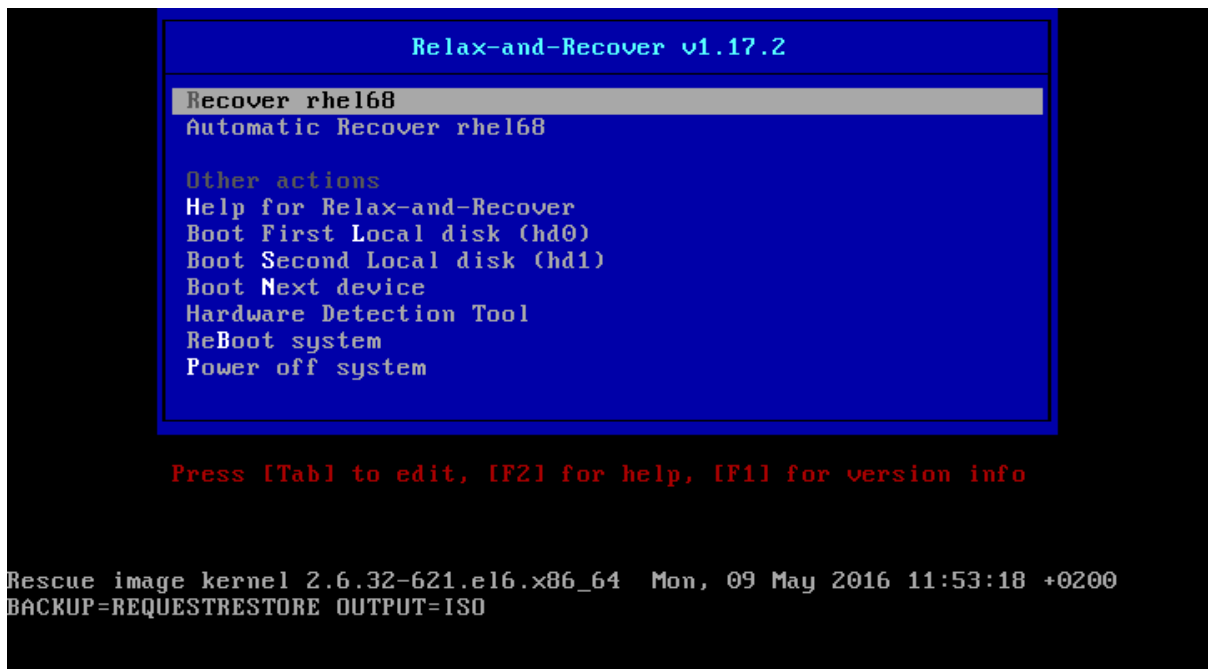


Figure 34.1. Rescue system: menu

3. You are taken to the prompt:

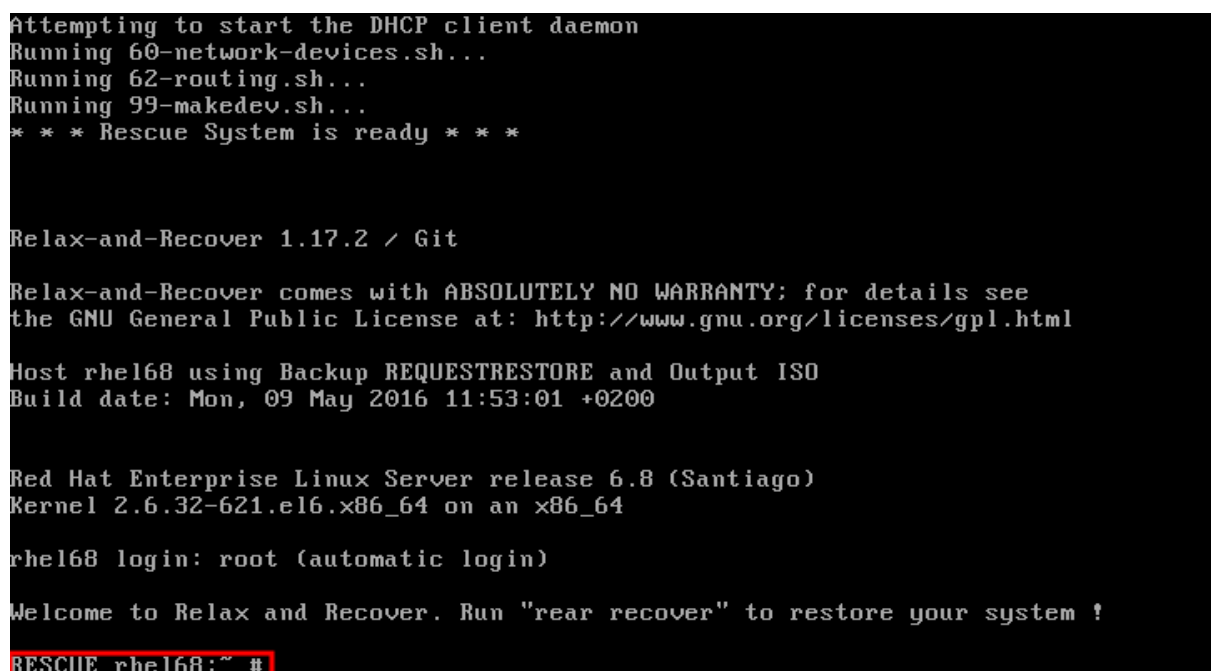


Figure 34.2. Rescue system: prompt

**WARNING**

Once you have started recovery in the next step, it probably cannot be undone and you may lose anything stored on the physical disks of the system.

4. Run the **rear recover** command to perform the restore or migration. The rescue system then recreates the partition layout and filesystems:

```

RESCUE rhel68:~ # rear recover
Relax-and-Recover 1.17.2 / Git
Using log file: /var/log/rear/rear-rhel68.log
NOTICE: Will do driver migration
Comparing disks.
Disk configuration is identical, proceeding with restore.
Start system layout restoration.
Creating partitions for disk /dev/sda (msdos)
Creating LUM PU /dev/sda2
Restoring LUM VG VolGroup
Sleeping 3 seconds to let udev or systemd-udev create their devices...
Creating ext4-filesystem / on /dev/mapper/VolGroup-lv_root
Mounting filesystem /
Creating ext4-filesystem /boot on /dev/sda1
Mounting filesystem /boot
Creating swap on /dev/mapper/VolGroup-lv_swap
Disk layout created.
Please start the restore process on your backup host.

Make sure that you restore the data into '/mnt/local' instead of '/' because the
hard disks of the recovered system are mounted there.

Please restore your backup in the provided shell and, when finished, type exit
in the shell to continue recovery.
rear> _

```

**Figure 34.3. Rescue system: running "rear recover"**

5. Restore user and system files from the backup into the **/mnt/local/** directory.

**Example 34.3. Restoring User and System Files**

In this example, the backup file is a **tar** archive created per instructions in [Section 34.2.1.1, "Configuring the Internal Backup Method"](#). First, copy the archive from its storage, then unpack the files into **/mnt/local/**, then delete the archive:

```

~]# scp root@192.168.122.6:/srv/backup/rhel68/backup.tar.gz
/mnt/local/
~]# tar xf /mnt/local/backup.tar.gz -C /mnt/local/
~]# rm -f /mnt/local/backup.tar.gz

```

The new storage has to have enough space both for the archive and the extracted files.

6. Verify that the files have been restored:

```

~]# ls /mnt/local/

```

```

Mounting filesystem /
Creating ext4-filesystem /boot on /dev/sda1
Mounting filesystem /boot
Creating swap on /dev/mapper/vg_rhel68-lv_swap
Disk layout created.
Please start the restore process on your backup host.

Make sure that you restore the data into '/mnt/local' instead of '/' because the
hard disks of the recovered system are mounted there.

Please restore your backup in the provided shell and, when finished, type exit
in the shell to continue recovery.
rear> scp -r root@192.168.122.6:/srv/backup/rhel68/backup.tar.gz /mnt/local/
The authenticity of host '192.168.122.6 (192.168.122.6)' can't be established.
RSA key fingerprint is c6:b0:9b:52:88:5a:c5:a1:3d:4c:40:ed:7a:88:33:77.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.122.6' (RSA) to the list of known hosts.
root@192.168.122.6's password:
backup.tar.gz                               100% 736MB 18.0MB/s 00:41
rear> tar xf /mnt/local/backup.tar.gz -C /mnt/local/
rear> rm -f /mnt/local/backup.tar.gz
rear> ls /mnt/local
bin      cgroup  etc     lib     lost+found  misc  net  proc  sbin  srv  tmp  var
boot    dev     home   lib64  media      mnt   opt  root  selinux  sys  usr
rear>

```

Figure 34.4. Rescue system: restoring user and system files from the backup

7. Ensure that SELinux relabels the files on the next boot:

```
~]# touch /mnt/local/.autorelabel
```

Otherwise you may be unable to log in the system, because the `/etc/passwd` file may have the incorrect SELinux context.

8. Finish the recovery and reboot the system:

```

rear> scp -r root@192.168.122.6:/srv/backup/rhel68/backup.tar.gz /mnt/local/
The authenticity of host '192.168.122.6 (192.168.122.6)' can't be established.
RSA key fingerprint is c6:b0:9b:52:88:5a:c5:a1:3d:4c:40:ed:7a:88:33:77.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.122.6' (RSA) to the list of known hosts.
root@192.168.122.6's password:
backup.tar.gz                               100% 736MB 18.0MB/s 00:41
rear> tar xf /mnt/local/backup.tar.gz -C /mnt/local/
rear> rm -f /mnt/local/backup.tar.gz
rear> ls /mnt/local
bin      cgroup  etc     lib     lost+found  misc  net  proc  sbin  srv  tmp  var
boot    dev     home   lib64  media      mnt   opt  root  selinux  sys  usr
rear> touch /mnt/local/.autorelabel
rear> exit
Did you restore the backup to /mnt/local? Are you ready to continue recovery?
y
exit
Updated initramfs with new drivers for Kernel 2.6.32-642.el6.x86_64.
Installing GRUB boot loader
Updating udev configuration (70-persistent-net.rules)
Updating udev configuration (70-persistent-cd.rules)

Finished recovering your system. You can explore it under '/mnt/local'.

RESCUE rhel68:~ # reboot

```

Figure 34.5. Rescue system: finishing recovery

ReaR will then reinstall the boot loader. Upon reboot, SELinux will relabel the whole filesystem.

Then you will be able to log in to the recovered system.

## 34.2. INTEGRATING REAR WITH BACKUP SOFTWARE

The main purpose of ReaR is to produce a rescue system, but it can also be integrated with backup software. What integration means is different for the built-in, supported, and unsupported backup methods.

### 34.2.1. The Built-in Backup Method

ReaR ships with a built-in, or internal, backup method. This method is fully integrated with ReaR, which has these advantages:

- a rescue system and a full-system backup can be created using a single **rear mkbackup** command
- the rescue system restores files from the backup automatically

As a result, ReaR can cover the whole process of creating both the rescue system and the full-system backup.

#### 34.2.1.1. Configuring the Internal Backup Method

To make ReaR use its internal backup method, add these lines to `/etc/rear/local.conf`:

```
BACKUP=NETFS
BACKUP_URL=backup location
```

These lines configure ReaR to create an archive with a full-system backup using the **tar** command. Substitute *backup location* with one of the options from the "Backup Software Integration" section of the `rear(8)` man page. Make sure that the backup location has enough space.

#### Example 34.4. Adding tar Backups

To expand the example in [Section 34.1, "Basic ReaR Usage"](#), configure ReaR to also output a **tar** full-system backup into the `/srv/backup/` directory:

```
OUTPUT=ISO
OUTPUT_URL=file:///mnt/rescue_system/
BACKUP=NETFS
BACKUP_URL=file:///srv/backup/
```

The internal backup method allows further configuration.

- To keep old backup archives when new ones are created, add this line:

```
NETFS_KEEP_OLD_BACKUP_COPY=y
```

- By default, ReaR creates a full backup on each run. To make the backups incremental, meaning that only the changed files are backed up on each run, add this line:

```
BACKUP_TYPE=incremental
```

This automatically sets **NETFS\_KEEP\_OLD\_BACKUP\_COPY** to **y**.

- To ensure that a full backup is done regularly in addition to incremental backups, add this line:

```
FULLBACKUPDAY="Day"
```

Substitute *Day* with one of the "Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun".

- ReaR can also include both the rescue system and the backup in the ISO image. To achieve this, set the **BACKUP\_URL** directive to **iso:///backup/**:

```
BACKUP_URL=iso:///backup/
```

This is the simplest method of full-system backup, because the rescue system does not need the user to fetch the backup during recovery. However, it needs more storage. Also, single-ISO backups cannot be incremental.



#### NOTE

Currently ReaR creates two copies of the ISO image, thus consuming two times more storage. For more information, see note *ReaR creates two ISO images instead of one* in [Red Hat Enterprise Linux 6 Release Notes](#).

#### Example 34.5. Configuring Single-ISO Rescue System and Backups

This configuration creates a rescue system and a backup file as a single ISO image and puts it into the **/srv/backup/** directory:

```
OUTPUT=ISO
OUTPUT_URL=file:///srv/backup/
BACKUP=NETFS
BACKUP_URL=iso:///backup/
```

- To use **rsync** instead of **tar**, add this line:

```
BACKUP_PROG=rsync
```

Note that incremental backups are only supported when using **tar**.

#### 34.2.1.2. Creating a Backup Using the Internal Backup Method

With **BACKUP=NETFS** set, ReaR can create either a rescue system, a backup file, or both.

- To create *a rescue system only*, run:

```
rear mkrescue
```

- To create *a backup only*, run:

```
rear mkbackuponly
```

- To create *a rescue system and a backup*, run:

```
rear mkbbackup
```

Note that triggering backup with ReaR is only possible if using the NETFS method. ReaR cannot trigger other backup methods.



#### NOTE

When restoring, the rescue system created with the **BACKUP=NETFS** setting expects the backup to be present before executing **rear recover**. Hence, once the rescue system boots, copy the backup file into the directory specified in **BACKUP\_URL**, unless using a single ISO image. Only then run **rear recover**.

To avoid recreating the rescue system unnecessarily, you can check whether storage layout has changed since the last rescue system was created using these commands:

```
~]# rear checklayout
~]# echo $?
```

Non-zero status indicates a change in disk layout. Non-zero status is also returned if ReaR configuration has changed.



#### IMPORTANT

The **rear checklayout** command does not check whether a rescue system is currently present in the output location, and can return 0 even if it is not there. So it does not guarantee that a rescue system is available, only that the layout has not changed since the last rescue system has been created.

#### Example 34.6. Using rear checklayout

To create a rescue system, but only if the layout has changed, use this command:

```
~]# rear checklayout || rear mkrescue
```

### 34.2.2. Supported Backup Methods

In addition to the NETFS internal backup method, ReaR supports several external backup methods. This means that the rescue system restores files from the backup automatically, but the backup creation cannot be triggered using ReaR.

For a list and configuration options of the supported external backup methods, see the "Backup Software Integration" section of the `rear(8)` man page.

### 34.2.3. Unsupported Backup Methods

With unsupported backup methods, there are two options:

1. The rescue system prompts the user to manually restore the files. This scenario is the one described in "Basic ReaR Usage", except for the backup file format, which may take a different form than a **tar** archive.



2. ReaR executes the custom commands provided by the user. To configure this, set the **BACKUP** directive to **EXTERNAL**. Then specify the commands to be run during backing up and restoration using the **EXTERNAL\_BACKUP** and **EXTERNAL\_RESTORE** directives. Optionally, also specify the **EXTERNAL\_IGNORE\_ERRORS** and **EXTERNAL\_CHECK** directives. See `/usr/share/rear/conf/default.conf` for an example configuration.

## APPENDIX A. CONSISTENT NETWORK DEVICE NAMING

Red Hat Enterprise Linux 6 provides consistent network device naming for network interfaces. This feature changes the name of network interfaces on a system in order to make locating and differentiating the interfaces easier.

Traditionally, network interfaces in Linux are enumerated as **eth[0123...]**, but these names do not necessarily correspond to actual labels on the chassis. Modern server platforms with multiple network adapters can encounter non-deterministic and counter-intuitive naming of these interfaces. This affects both network adapters embedded on the motherboard (*Lan-on-Motherboard*, or *LOM*) and add-in (single and multiport) adapters.

The new naming convention assigns names to network interfaces based on their physical location, whether embedded or in PCI slots. By converting to this naming convention, system administrators will no longer have to guess at the physical location of a network port, or modify each system to rename them into some consistent order.

This feature, implemented via the **biosdevname** program, will change the name of all embedded network interfaces, PCI card network interfaces, and virtual function network interfaces from the existing **eth[0123...]** to the new naming convention as shown in [Table A.1, “The new naming convention”](#).

**Table A.1. The new naming convention**

Device	Old Name	New Name
Embedded network interface (LOM)	<b>eth[0123...]</b>	<b>em[1234...]</b> <sup>[a]</sup>
PCI card network interface	<b>eth[0123...]</b>	<b>p&lt;slot&gt;p&lt;ethernet port&gt;</b> <sup>[b]</sup>
Virtual function	<b>eth[0123...]</b>	<b>p&lt;slot&gt;p&lt;ethernet port&gt;_&lt;virtual interface&gt;</b> <sup>[c]</sup>
<p>[a] New enumeration starts at <b>1</b>.</p> <p>[b] For example: <b>p3p4</b></p> <p>[c] For example: <b>p3p4_1</b></p>		

System administrators may continue to write rules in **/etc/udev/rules.d/70-persistent-net.rules** to change the device names to anything desired; those will take precedence over this physical location naming convention.

### A.1. AFFECTED SYSTEMS

Consistent network device naming is enabled by default for a set of **Dell PowerEdge, C Series**, and **Precision Workstation** systems. For more details regarding the impact on Dell systems, visit <https://access.redhat.com/kb/docs/DOC-47318>.

For all other systems, it will be disabled by default; see [Section A.2, “System Requirements”](#) and [Section A.3, “Enabling and Disabling the Feature”](#) for more details.

Regardless of the type of system, Red Hat Enterprise Linux 6 guests running under Red Hat Enterprise Linux 5 hosts will not have devices renamed, since the virtual machine BIOS does not provide SMBIOS information. Upgrades from Red Hat Enterprise Linux 6.0 to Red Hat Enterprise Linux 6.1 are unaffected, and the old `eth[0123...]` naming convention will continue to be used.

## A.2. SYSTEM REQUIREMENTS

The `biosdevname` program uses information from the system's BIOS, specifically the *type 9* (System Slot) and *type 41* (Onboard Devices Extended Information) fields contained within the SMBIOS. If the system's BIOS does not have SMBIOS version 2.6 or higher and this data, the new naming convention will not be used. Most older hardware does not support this feature because of a lack of BIOSes with the correct SMBIOS version and field information. For BIOS or SMBIOS version information, contact your hardware vendor.

For this feature to take effect, the `biosdevname` package must also be installed. The `biosdevname` package is part of the `base` package group in Red Hat Enterprise Linux 6. All install options, except for **Minimal Install**, include this package. It is not installed on upgrades of Red Hat Enterprise Linux 6.0 to RHEL 6.1.

## A.3. ENABLING AND DISABLING THE FEATURE

To disable the consistent network device naming on Dell systems that would normally have it on by default, pass the following option on the boot command line, both during and after installation:

```
biosdevname=0
```

To enable this feature on other system types that meet the minimum requirements (see [Section A.2, "System Requirements"](#)), pass the following option on the boot command line, both during and after installation:

```
biosdevname=1
```

Unless the system meets the minimum requirements, this option will be ignored and the system will boot with the traditional network interface name format.

If the `biosdevname` install option is specified, it must remain as a boot option for the lifetime of the system.

## A.4. NOTES FOR ADMINISTRATORS

Many system customization files can include network interface names, and thus will require updates if moving a system from the old convention to the new convention. If you use the new naming convention, you will also need to update network interface names in areas such as custom iptables rules, scripts altering irqbalance, and other similar configuration files. Also, enabling this change for installation will require modification to existing kickstart files that use device names via the `ksdevice` parameter; these kickstart files will need to be updated to use the network device's MAC address or the network device's new name.

Red Hat strongly recommends that you consider this feature to be an install-time choice; enabling or disabling the feature post-install, while technically possible, can be complicated and is not recommended. For those system administrators who want to do so, on a system that meets the minimum requirements, remove the `/etc/udev/rules.d/70-persistent-net.rules` file and the `HWADDR` lines from all

**/etc/sysconfig/network-scripts/ifcfg-\*** files. In addition, rename those **ifcfg-\*** files to use this new naming convention. The new names will be in effect after reboot. Remember to update any custom scripts, iptables rules, and service configuration files that might include network interface names.

## APPENDIX B. RPM

The *RPM Package Manager* (RPM) is an open packaging system, which runs on Red Hat Enterprise Linux as well as other Linux and UNIX systems. Red Hat, Inc. 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 to work with the *RPM format*. RPM is itself provided as a 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.

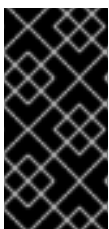
The RPM package format has been improved for Red Hat Enterprise Linux 6. RPM packages are now compressed using the XZ lossless data compression format, which has the benefit of greater compression and less CPU usage during decompression, and support multiple strong hash algorithms, such as SHA-256, for package signing and verification.



### 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 resolution, and will complain and force system integrity checks if you use RPM as well to install and remove packages. For these reasons, it is highly recommended that you use **Yum** instead of RPM whenever possible to perform package management tasks. See [Chapter 8, 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. See [Chapter 9, PackageKit](#) for details.



### 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. Many of the following examples show RPM packages compiled for the AMD64/Intel 64 computer architectures; thus, the RPM file name ends in **x86\_64.rpm**.

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 makes changes to your system, you must be logged in as root to install, remove, or upgrade an RPM package.

## B.1. RPM DESIGN GOALS

To understand how to use RPM, it can be helpful to understand the design goals of RPM:

### Upgradability

With RPM, you can upgrade individual components of your system without completely reinstalling. 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 your machine (as you might need to with operating systems based on other packaging systems). RPM allows 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 entire database for packages or even just certain files. You can also easily find out what package a file belongs to and from where the package came. 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. If you are worried that you deleted an important file for some package, you can verify the package. You are then notified of anomalies, if any—at which point you can reinstall the package, if necessary. Any configuration files that you modified are preserved during reinstallation.

### Pristine Sources

A crucial design goal was 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, too.

## B.2. USING RPM

RPM has five basic modes of operation (not counting 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 `man rpm`. You can also see [Section B.5, “Additional Resources”](#) for more information on RPM.

## B.2.1. Finding RPM Packages

Before using any RPM packages, you must know where to find them. An Internet search returns many RPM repositories, but if you are looking for Red Hat RPM packages, they can be found at the following locations:

- The Red Hat Enterprise Linux installation media contain many installable RPMs.
- The initial RPM repositories provided with the YUM package manager. See [Chapter 8, Yum](#) for details on how to use the official Red Hat Enterprise Linux package repositories.
- The Extra Packages for Enterprise Linux (EPEL) is a community effort to provide 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.

- The Red Hat Errata Page, available at <http://www.redhat.com/apps/support/errata/>.

## B.2.2. Installing and Upgrading

RPM packages typically have file names like `tree-1.5.3-2.e16.x86_64.rpm`. The file name includes the package name (`tree`), version (`1.5.3`), release (`2`), operating system major version (`e16`) and CPU architecture (`x86_64`).

You can use `rpm`'s `-U` option to:

- upgrade an existing but older package on the system to a newer version, or
- install the package even if an older version is not already installed.

That is, `rpm -U <rpm_file>` is able to perform the function of either *upgrading* or *installing* as is appropriate for the package.

Assuming the `tree-1.5.3-2.e16.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 as determined by `rpm`:

```
rpm -Uvh tree-1.5.3-2.e16.x86_64.rpm
```



### NOTE

The `-v` and `-h` options (which are combined with `-U`) cause `rpm` to print more verbose output and display a progress meter using hash signs.

If the upgrade/installation is successful, the following output is displayed:

```
Preparing... #####
[100%]
  1:tree #####
[100%]
```



## WARNING

**rpm** provides two different options for installing packages: the aforementioned **-U** option (which historically stands for *upgrade*), and the **-i** option, historically standing for *install*. Because the **-U** option subsumes both install and upgrade functions, we recommend to use **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 30, 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. For example, if the verification of the signature fails, an error message such as the following is displayed:

```
error: tree-1.5.3-2.el6.x86_64.rpm: Header V3 RSA/SHA256 signature: BAD,
key ID
d22e77f2
```

If it is a new, header-only, signature, an error message such as the following is displayed:

```
error: tree-1.5.3-2.el6.x86_64.rpm: Header V3 RSA/SHA256 signature: BAD,
key ID d22e77f2
```

If you do not have the appropriate key installed to verify the signature, the message contains the word **NOKEY**:

```
warning: tree-1.5.3-2.el6.x86_64.rpm: Header V3 RSA/SHA1 signature: NOKEY,
key ID 57bbccba
```

See [Section B.3, “Checking a Package's Signature”](#) for more information on checking a package's signature.

### B.2.2.1. Package Already Installed

If a package of the same name and version is already installed, the following output is displayed:



```
Preparing... #####
[100%]
package tree-1.5.3-2.el6.x86_64 is already installed
```

However, if you want to install the package anyway, you can use the **--replacepks** option, which tells RPM to ignore the error:

```
rpm -Uvh --replacepks tree-1.5.3-2.el6.x86_64.rpm
```

This option is helpful if files installed from the RPM were deleted or if you want the original configuration files from the RPM to be installed.

### B.2.2.2. Conflicting Files

If you attempt to install a package that contains a file which has already been installed by another package, the following is displayed:

```
Preparing... #####
file /usr/bin/foobar from install of foo-1.0-1.el6.x86_64 conflicts
with file from package bar-3.1.1.el6.x86_64
```

To make RPM ignore this error, use the **--replacefiles** option:

```
rpm -Uvh --replacefiles foo-1.0-1.el6.x86_64.rpm
```

### B.2.2.3. Unresolved Dependency

RPM packages may 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 which has an unresolved dependency, output similar to the following is displayed:

```
error: Failed dependencies:
bar.so.3()(64bit) is needed by foo-1.0-1.el6.x86_64
```

If you are installing a package from the Red Hat Enterprise Linux installation media, such as from a CD-ROM or DVD, the dependencies may be available. 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 command:

```
rpm -Uvh foo-1.0-1.el6.x86_64.rpm bar-3.1.1.el6.x86_64.rpm
```

If installation of both packages is successful, output similar to the following is displayed:

```
Preparing... #####
[100%]
 1:foo ##### [
50%]
 2:bar #####
[100%]
```

You can try the **--whatprovides** option to determine which package contains the required file.

```
rpm -q --whatprovides "bar.so.3"
```

If the package that contains **bar.so.3** is in the RPM database, the name of the package is displayed:

```
bar-3.1.1.e16.i586.rpm
```



### WARNING

Although we can *force* **rpm** to install a package that gives us a **Failed dependencies** error (using the **--nodeps** option), this is *not* recommended, and will usually result in the installed package failing to run. Installing or removing packages with **rpm --nodeps** can cause applications to misbehave and/or crash, and can cause serious package management problems or, possibly, system failure. For these reasons, it is best to heed such warnings; the package manager—whether **RPM**, **Yum** or **PackageKit**—shows us these warnings and suggests possible fixes because accounting for dependencies is critical. The **Yum** package manager can perform dependency resolution and fetch dependencies from online repositories, making it safer, easier and smarter than forcing **rpm** to carry out actions without regard to resolving dependencies.

### B.2.3. Configuration File Changes

Because RPM performs intelligent upgrading of packages with configuration files, you may see one or the other of the following messages:

```
saving /etc/foo.conf as /etc/foo.conf.rpmsave
```

This message means that 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, **foo.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 with a **diff** program.

If you attempt to upgrade to a package with an *older* version number (that is, if a higher version of the package is already installed), the output is similar to the following:

```
package foo-2.0-1.e16.x86_64.rpm (which is newer than foo-1.0-1) is
already installed
```

To force RPM to upgrade anyway, use the **--oldpackage** option:

```
rpm -Uvh --oldpackage foo-1.0-1.e16.x86_64.rpm
```

## B.2.4. Uninstalling

Uninstalling a package is just as simple as installing one. Type the following command at a shell prompt:

```
rpm -e foo
```



### NOTE

Notice that we used the package *name* **foo**, not the name of the original package *file*, **foo-1.0-1.el6.x86\_64**. If you attempt to uninstall a package using the **rpm -e** command and the original full file name, you will 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:
    libgs.so.8()(64bit) is needed by (installed) libspectre-0.2.2-
3.el6.x86_64
    libgs.so.8()(64bit) is needed by (installed) foomatic-4.0.3-
1.el6.x86_64
    libijs-0.35.so()(64bit) is needed by (installed) gutenprint-5.2.4-
5.el6.x86_64
    ghostscript is needed by (installed) printer-filters-1.1-
4.el6.noarch
```

Similar to how we searched for a shared object library (i.e. a **<library\_name>.so.<number>** file) in [Section B.2.2.3, “Unresolved Dependency”](#), we can search for a 64-bit shared object library using this exact syntax (and making sure to quote the file name):

```
~]# rpm -q --whatprovides "libgs.so.8()(64bit)"
ghostscript-8.70-1.el6.x86_64
```



### WARNING

Although we can *force* **rpm** to remove a package that gives us a **Failed dependencies** error (using the **--nodeps** option), this is *not* recommended, and may cause harm to other installed applications. Installing or removing packages with **rpm --nodeps** can cause applications to misbehave and/or crash, and can cause serious package management problems or, possibly, system failure. For these reasons, it is best to heed such warnings; the package manager—whether **RPM**, **Yum** or **PackageKit**—shows us these warnings and suggests possible fixes because accounting for dependencies is critical. The **Yum** package manager can perform dependency resolution and fetch dependencies from online repositories, making it safer, easier and smarter than forcing **rpm** to carry out actions without regard to resolving dependencies.

## B.2.5. Freshening

Freshening is similar to upgrading, except that only existent packages are upgraded. Type the following command at a shell prompt:

```
rpm -Fvh foo-2.0-1.e16.x86_64.rpm
```

RPM's `freshen` option checks the versions of the packages specified on the command line against the versions of packages that have already been installed on your system. When a newer version of an already-installed package is processed by RPM's `freshen` option, it is upgraded to the newer version. However, RPM's `freshen` option does not install a package if no previously-installed package of the same name exists. This differs from RPM's `upgrade` option, as an `upgrade` *does* install packages whether or not an older version of the package was already installed.

Freshening works for single packages or package groups. If you have just downloaded a large number of different packages, and you only want to upgrade those packages that are already installed on your system, freshening does the job. Thus, you do not have to delete any unwanted packages from the group that you downloaded before using RPM.

In this case, issue the following with the `*.rpm` glob:

```
rpm -Fvh *.rpm
```

RPM then automatically upgrades only those packages that are already installed.

## B.2.6. Querying

The RPM database stores information about all RPM packages installed in your system. It is stored in the directory `/var/lib/rpm/`, and is used to query what packages are installed, what versions each package is, and to calculate any changes to any files in the package since installation, among other use cases.

To query this database, use the `-q` option. The `rpm -q package name` command displays the package name, version, and release number of the installed package `<package_name>`. For example, using `rpm -q tree` to query installed package `tree` might generate the following output:

```
tree-1.5.2.2-4.e16.x86_64
```

You can also use the following *Package Selection Options* (which is a subheading in the RPM man page: see `man rpm` for details) to further refine or qualify your query:

- `-a` — queries all currently installed packages.
- `-f <file_name>` — queries the RPM database for which package owns `<file_name>`. Specify the absolute path of the file (for example, `rpm -qf /bin/ls` instead of `rpm -qf ls`).
- `-p <package_file>` — queries the uninstalled package `<package_file>`.

There are a number of ways to specify what information to display about queried packages. The following options are used to select the type of information for which you are searching. These are called the *Package Query Options*.

- `-i` displays package information including name, description, release, size, build date, install date, vendor, and other miscellaneous information.

- **-l** displays the list of files that the package contains.
- **-s** displays the state of all the files in the package.
- **-d** displays a list of files marked as documentation (man pages, info pages, READMEs, etc.) in the package.
- **-c** displays a list of files marked as configuration files. These are the files you edit after installation to adapt and customize the package to your system (for example, **sendmail.cf**, **passwd**, **inittab**, etc.).

For options that display lists of files, add **-v** to the command to display the lists in a familiar **ls -l** format.

### B.2.7. Verifying

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 group of each file.

The command **rpm -V** verifies a package. You can use any of the *Verify Options* listed for querying to specify the packages you want to verify. A simple use of verifying is **rpm -V tree**, which verifies that all the files in the tree package are as they were when they were originally installed. For example:

- To verify a package containing a particular file:

```
rpm -vf /usr/bin/tree
```

In this example, **/usr/bin/tree** is the absolute path to the file used to query a package.

- To verify ALL installed packages throughout the system (which will take some time):

```
rpm -Va
```

- To verify an installed package against an RPM package file:

```
rpm -Vp tree-1.5.3-2.e16.x86_64.rpm
```

This command can be useful if you suspect that your RPM database is corrupt.

If everything verified properly, there is no output. If there are any discrepancies, they are displayed. The format of the output is a string of eight characters (a "c" denotes a configuration file) and then the file name. Each of the eight characters denotes the result of a comparison of one attribute of the file to the value of that attribute recorded in the RPM database. A single period (.) means the test passed. The following characters denote specific discrepancies:

- **5** — MD5 checksum
- **S** — file size
- **L** — symbolic link
- **T** — file modification time

- **D** — device
- **U** — user
- **G** — group
- **M** — mode (includes permissions and file type)
- **?** — unreadable file (file permission errors, for example)

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.

## B.3. CHECKING A PACKAGE'S SIGNATURE

If you want to verify that a package has not been corrupted or tampered with, examine only the md5sum by typing the following command at a shell prompt (where `<rpm_file>` is the file name of the RPM package):

```
rpm -K --nosignature <rpm_file>
```

The message `<rpm_file>: rsa sha1 (md5) pgp md5 OK` (specifically the *OK* part of it) is displayed. This brief message means that the file was not corrupted during download. To see a more verbose message, replace `-K` with `-Kvv` in the command.

On the other hand, how trustworthy is the developer who created the package? If the package is *signed* with the developer's GnuPG *key*, you know that the developer really is who they say they are.

An RPM package can be signed using *GNU Privacy Guard* (or GnuPG), to help you make certain your downloaded package is trustworthy.

GnuPG is a tool for secure communication; it is a complete and free replacement for the encryption technology of PGP, an electronic privacy program. With GnuPG, you can authenticate the validity of documents and encrypt/decrypt data to and from other recipients. GnuPG is capable of decrypting and verifying PGP 5.x files as well.

During installation, GnuPG is installed by default. That way you can immediately start using GnuPG to verify any packages that you receive from Red Hat. Before doing so, you must first import Red Hat's public key.

### B.3.1. Importing Keys

To verify Red Hat packages, you must import the Red Hat GnuPG key. To do so, execute the following command at a shell prompt:

```
rpm --import /usr/share/rhn/RPM-GPG-KEY
```

To display a list of all keys installed for RPM verification, execute the command:

```
rpm -qa gpg-pubkey*
```

For the Red Hat key, the output includes:

```
gpg-pubkey-db42a60e-37ea5438
```

To display details about a specific key, use `rpm -qi` followed by the output from the previous command:

```
rpm -qi gpg-pubkey-db42a60e-37ea5438
```

### B.3.2. Verifying Signature of Packages

To check the GnuPG signature of an RPM file after importing the builder's GnuPG key, use the following command (replace `<rpm-file>` with the file name of the RPM package):

```
rpm -K <rpm-file>
```

If all goes well, the following message is displayed: **md5 gpg OK**. This means that the signature of the package has been verified, that it is not corrupt, and therefore is safe to install and use.

## B.4. PRACTICAL AND COMMON EXAMPLES OF RPM USAGE

RPM is a useful tool for both managing your system and diagnosing and fixing problems. The best way to make sense of all its options is to look at some examples.

- Perhaps you have deleted some files by accident, but you are not sure what you deleted. To verify your entire system and see what might be missing, you could try the following command:

```
rpm -Va
```

If some files are missing or appear to have been corrupted, you should probably either re-install the package or uninstall and then re-install the package.

- At some point, you might see a file that you do not recognize. To find out which package owns it, enter:

```
rpm -qf /usr/bin/ghostscript
```

The output would look like the following:

```
ghostscript-8.70-1.e16.x86_64
```

- We can combine the above two examples in the following scenario. Say you are having problems with `/usr/bin/paste`. You would like to verify the package that owns that program, but you do not know which package owns `paste`. Enter the following command,

```
rpm -vf /usr/bin/paste
```

and the appropriate package is verified.

- Do you want to find out more information about a particular program? You can try the following command to locate the documentation which came with the package that owns that program:

```
rpm -qdf /usr/bin/free
```

The output would be similar to the following:

■

```

/usr/share/doc/procps-3.2.8/BUGS
/usr/share/doc/procps-3.2.8/FAQ
/usr/share/doc/procps-3.2.8/NEWS
/usr/share/doc/procps-3.2.8/TODO
/usr/share/man/man1/free.1.gz
/usr/share/man/man1/pgrep.1.gz
/usr/share/man/man1/pkill.1.gz
/usr/share/man/man1/pmap.1.gz
/usr/share/man/man1/ps.1.gz
/usr/share/man/man1/pwdx.1.gz
/usr/share/man/man1/skill.1.gz
/usr/share/man/man1/slabtop.1.gz
/usr/share/man/man1/snice.1.gz
/usr/share/man/man1/tload.1.gz
/usr/share/man/man1/top.1.gz
/usr/share/man/man1/uptime.1.gz
/usr/share/man/man1/w.1.gz
/usr/share/man/man1/watch.1.gz
/usr/share/man/man5/sysctl.conf.5.gz
/usr/share/man/man8/sysctl.8.gz
/usr/share/man/man8/vmstat.8.gz

```

- You may find a new RPM, but you do not know what it does. To find information about it, use the following command:

```
rpm -qip crontabs-1.10-32.1.el6.noarch.rpm
```

The output would be similar to the following:

```

Name           : crontabs                      Relocations: (not
relocatable)
Version        : 1.10                          Vendor: Red Hat,
Inc.
Release        : 32.1.el6                      Build Date: Thu 03 Dec
2009 02:17:44 AM CET
Install Date: (not installed)                  Build Host: js20-bc1-
11.build.redhat.com
Group          : System Environment/Base        Source RPM: crontabs-
1.10-32.1.el6.src.rpm
Size           : 2486                          License: Public
Domain and GPLv2
Signature      : RSA/8, Wed 24 Feb 2010 08:46:13 PM CET, Key ID
938a80caf21541eb
Packager       : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
Summary       : Root crontab files used to schedule the execution of
programs
Description   :
The crontabs package contains root crontab files and directories.
You will need to install cron daemon to run the jobs from the
crontabs.
The cron daemon such as croneie or fcron checks the crontab files to
see when particular commands are scheduled to be executed. If
commands
are scheduled, it executes them.

```



```
Crontabs handles a basic system function, so it should be installed
on
your system.
```

- Perhaps you now want to see what files the **crontabs** RPM package installs. You would enter the following:

```
rpm -qpl crontabs-1.10-32.1.el6.noarch.rpm
```

The output is similar to the following:

```
/etc/cron.daily
/etc/cron.hourly
/etc/cron.monthly
/etc/cron.weekly
/etc/crontab
/usr/bin/run-parts
/usr/share/man/man4/crontabs.4.gz
```

These are just a few examples. As you use RPM, you may find more uses for it.

## B.5. ADDITIONAL RESOURCES

RPM is an extremely complex utility with many options and methods for querying, installing, upgrading, and removing packages. See the following resources to learn more about RPM.

### B.5.1. Installed Documentation

- **rpm --help** — This command displays a quick reference of RPM parameters.
- **man rpm** — The RPM man page gives more detail about RPM parameters than the **rpm --help** command.

### B.5.2. Useful Websites

- The RPM website — <http://www.rpm.org/>
- The RPM mailing list can be subscribed to, and its archives read from, here — <https://lists.rpm.org/mailman/listinfo/rpm-list>

## APPENDIX C. 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.

### C.1. THE X SERVER

Red Hat Enterprise Linux 6 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 C.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.

### C.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 the rudimentary *Tab Window Manager*

(*twm*) to the highly developed and interactive desktop environment (such as *GNOME* or *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*.

### C.2.1. Maximum number of concurrent GUI sessions

Multiple GUI sessions for different users can be run at the same time on the same machine. The maximum number of concurrent GUI sessions is limited by the hardware, especially by the memory size, and by the workload demands of the running applications. For common PCs the maximum possible number of concurrent GUI sessions is not higher than 10 to 15, depending on previously described circumstances. Logging the same user into GNOME more than once on the same machine is not supported, because some applications could terminate unexpectedly.

### C.2.2. 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+ 2 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+ 2 and the Qt libraries are present, KDE applications can run in GNOME and vice versa.

### C.2.3. 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 five different window managers.

#### **metacity**

The *Metacity* window manager is the default window manager for GNOME. It is a simple and efficient window manager which supports custom themes. This window manager is automatically pulled in as a dependency when the GNOME desktop is installed.

#### **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.

## compiz

The *Compiz* compositing window manager is based on OpenGL and can use 3D graphics hardware to create fast compositing desktop effects for window management. Advanced features, such as a cube workspace, are implemented as loadable plug-ins. To run this window manager, you need to install the `compiz` package.

## 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.

## twm

The minimalist *Tab Window Manager* (`twm`), which provides the most basic tool set among the available window managers, can be used either as a stand-alone or with a desktop environment. To run this window manager, you need to install the `xorg-x11-twm` package.

## C.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.



### NOTE

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 6 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 `vmouse` driver configured in the `xorg.conf` file are, by default, ignored by the X server. See [Section C.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.

### C.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
```

### C.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 C.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.

### C.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

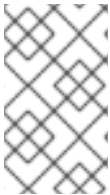
`/etc/X11/xorg.conf` file. More detailed information about the X server configuration file can be found in the `xorg.conf(5)` man page. This section is mostly intended for advanced users as most configuration options described below are not needed in typical configuration scenarios.

### C.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.



#### NOTE

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.

### C.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 6, 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.

### C.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 (see [Section C.3.3.4, “The ServerLayout Section”](#) for details).

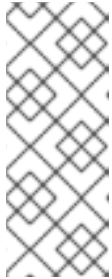
Each entry within the **ServerFlags** section occupies a single line and begins with the term **Option** followed by an option enclosed in double quotation marks (").

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.



#### NOTE

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 C.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.

### C.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 C.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.

### C.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.

### C.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.

### C.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 **"dpms"** (for Display Power Management Signaling, a VESA standard), which activates the Energy Star energy compliance setting for the monitor.

### C.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.

### C.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/>.

## C.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 *X FreeType 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.



#### NOTE

Fontconfig uses the `/etc/fonts/fonts.conf` configuration file, which should not be edited by hand.



#### WARNING

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.

### C.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/`).



#### NOTE

Individual users may also install fonts interactively, by typing `fonts:///` into the **Nautilus** address bar, and dragging the new font files there.

## C.5. RUNLEVELS AND X

In most cases, the Red Hat Enterprise Linux installer configures a machine to boot into a graphical login environment, known as *runlevel 5*. It is possible, however, to boot into a text-only multi-user mode called *runlevel 3* and begin an X session from there.

The following subsections review how X starts up in both runlevel 3 and runlevel 5. For more information about runlevels, see [Section 12.1, “Configuring the Default Runlevel”](#).

### C.5.1. Runlevel 3

When in runlevel 3, the best way to start an X session is to log in and type **startx**. The **startx** command is a front-end to the **xinit** command, which launches the X server (**Xorg**) and connects X client applications to it. Because the user is already logged into the system at runlevel 3, **startx** does not launch a display manager or authenticate users. See [Section C.5.2, “Runlevel 5”](#) for more information about display managers.

1. When the **startx** command is executed, it searches for the **.xinitrc** file in the user's home directory to define the desktop environment and possibly other X client applications to run. If no **.xinitrc** file is present, it uses the system default **/etc/X11/xinit/xinitrc** file instead.
2. The default **xinitrc** script then searches for user-defined files and default system files, including **.Xresources**, **.Xmodmap**, and **.Xkbmap** in the user's home directory, and **Xresources**, **Xmodmap**, and **Xkbmap** in the **/etc/X11/** directory. The **Xmodmap** and **Xkbmap** files, if they exist, are used by the **xmodmap** utility to configure the keyboard. The **Xresources** file is read to assign specific preference values to applications.
3. After setting the above options, the **xinitrc** script executes all scripts located in the **/etc/X11/xinit/xinitrc.d/** directory. One important script in this directory is **xinput.sh**, which configures settings such as the default language.
4. The **xinitrc** script attempts to execute **.Xclients** in the user's home directory and turns to **/etc/X11/xinit/Xclients** if it cannot be found. The purpose of the **Xclients** file is to start the desktop environment or, possibly, just a basic window manager. The **.Xclients** script in the user's home directory starts the user-specified desktop environment in the **.Xclients-default** file. If **.Xclients** does not exist in the user's home directory, the standard **/etc/X11/xinit/Xclients** script attempts to start another desktop environment, trying GNOME first, then KDE, followed by **twm**.

When in runlevel 3, the user is returned to a text mode user session after ending an X session.

### C.5.2. Runlevel 5

When the system boots into runlevel 5, a special X client application called a *display manager* is launched. A user must authenticate using the display manager before any desktop environment or window managers are launched.

Depending on the desktop environments installed on the system, three different display managers are available to handle user authentication.

- **GDM** (GNOME Display Manager) — The default display manager for Red Hat Enterprise Linux. **GNOME** allows the user to configure language settings, shutdown, restart or log in to the system.
- **KDM** — KDE's display manager which allows the user to shutdown, restart or log in to the system.

- **xdm** (X Window Display Manager) — A very basic display manager which only lets the user log in to the system.

When booting into runlevel 5, the `/etc/X11/prefdm` script determines the preferred display manager by referencing the `/etc/sysconfig/desktop` file. A list of options for this file is available in this file:

```
/usr/share/doc/initscripts-<version-number>/sysconfig.txt
```

where `<version-number>` is the version number of the initscripts package.

Each of the display managers reference the `/etc/X11/xdm/Xsetup_0` file to set up the login screen. Once the user logs into the system, the `/etc/X11/xdm/GiveConsole` script runs to assign ownership of the console to the user. Then, the `/etc/X11/xdm/Xsession` script runs to accomplish many of the tasks normally performed by the `xinitrc` script when starting X from runlevel 3, including setting system and user resources, as well as running the scripts in the `/etc/X11/xinit/xinitrc.d/` directory.

Users can specify which desktop environment they want to use when they authenticate using the **GNOME** or **KDE** display managers by selecting it from the **Sessions** menu item accessed by selecting **System** → **Preferences** → **More Preferences** → **Sessions**. If the desktop environment is not specified in the display manager, the `/etc/X11/xdm/Xsession` script checks the `.xsession` and `.Xclients` files in the user's home directory to decide which desktop environment to load. As a last resort, the `/etc/X11/xinit/Xclients` file is used to select a desktop environment or window manager to use in the same way as runlevel 3.

When the user finishes an X session on the default display (`:0`) and logs out, the `/etc/X11/xdm/TakeConsole` script runs and reassigns ownership of the console to the root user. The original display manager, which continues running after the user logged in, takes control by spawning a new display manager. This restarts the X server, displays a new login window, and starts the entire process over again.

The user is returned to the display manager after logging out of X from runlevel 5.

For more information on how display managers control user authentication, see the `/usr/share/doc/gdm-<version-number>/README`, where `<version-number>` is the version number for the gdm package installed, or the `xdm` man page.

## C.6. ACCESSING GRAPHICAL APPLICATIONS REMOTELY

It is possible to access graphical applications on a remote server using these methods:

- You can start a separate application directly from your SSH session in your local X server. For that, you need to enable X11 forwarding. See [Section 14.5.1, “X11 Forwarding”](#) for details.
- You can run the whole X session over network using VNC. This method can be useful, especially when you are using a workstation without X server, for example, a non-Linux system. See [Chapter 15, TigerVNC](#) for details.

## C.7. 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.

### C.7.1. 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.

### C.7.2. 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.



## APPENDIX D. THE SYSCONFIG DIRECTORY

This appendix outlines some of the files and directories found in the `/etc/sysconfig/` directory, their function, and their contents. The information in this appendix is not intended to be complete, as many of these files have a variety of options that are only used in very specific or rare circumstances.



### NOTE

The actual content of your `/etc/sysconfig/` directory depends on the programs you have installed on your machine. To find the name of the package the configuration file belongs to, type the following at a shell prompt:

```
~]$ yum provides /etc/sysconfig/filename
```

See [Section 8.2.4, “Installing Packages”](#) for more information on how to install new packages in Red Hat Enterprise Linux.

## D.1. FILES IN THE /ETC/SYSCONFIG/ DIRECTORY

The following sections offer descriptions of files normally found in the `/etc/sysconfig/` directory.

### D.1.1. /etc/sysconfig/arpwatch

The `/etc/sysconfig/arpwatch` file is used to pass arguments to the `arpwatch` daemon at boot time. By default, it contains the following option:

#### **OPTIONS=***value*

Additional options to be passed to the `arpwatch` daemon. For example:

```
OPTIONS="-u arpwatch -e root -s 'root (Arpwatch)'"
```

### D.1.2. /etc/sysconfig/authconfig

The `/etc/sysconfig/authconfig` file sets the authorization to be used on the host. By default, it contains the following options:

#### **USEMKHOMEDIR=***boolean*

A Boolean to enable (**yes**) or disable (**no**) creating a home directory for a user on the first login. For example:

```
USEMKHOMEDIR=no
```

#### **USEPAMACCESS=***boolean*

A Boolean to enable (**yes**) or disable (**no**) the PAM authentication. For example:

```
USEPAMACCESS=no
```

#### **USESSSDAUTH=***boolean*

A Boolean to enable (**yes**) or disable (**no**) the SSSD authentication. For example:

```
USESSSDAUTH=no
```

#### **USESHADOW=boolean**

A Boolean to enable (**yes**) or disable (**no**) shadow passwords. For example:

```
USESHADOW=yes
```

#### **USEWINBIND=boolean**

A Boolean to enable (**yes**) or disable (**no**) using Winbind for user account configuration. For example:

```
USEWINBIND=no
```

#### **USEDDB=boolean**

A Boolean to enable (**yes**) or disable (**no**) the FAS authentication. For example:

```
USEDDB=no
```

#### **USEFPRIND=boolean**

A Boolean to enable (**yes**) or disable (**no**) the fingerprint authentication. For example:

```
USEFPRIND=yes
```

#### **FORCESMARTCARD=boolean**

A Boolean to enable (**yes**) or disable (**no**) enforcing the smart card authentication. For example:

```
FORCESMARTCARD=no
```

#### **PASSWDALGORITHM=value**

The password algorithm. The *value* can be **bigcrypt**, **descrypt**, **md5**, **sha256**, or **sha512**. For example:

```
PASSWDALGORITHM=sha512
```

#### **USELDAPAUTH=boolean**

A Boolean to enable (**yes**) or disable (**no**) the LDAP authentication. For example:

```
USELDAPAUTH=no
```

#### **USELOCALAUTHORIZE=boolean**

A Boolean to enable (**yes**) or disable (**no**) the local authorization for local users. For example:

```
USELOCALAUTHORIZE=yes
```

**USECRACKLIB=*boolean***

A Boolean to enable (**yes**) or disable (**no**) using the CrackLib. For example:

```
USECRACKLIB=yes
```

**USEWINBINDAUTH=*boolean***

A Boolean to enable (**yes**) or disable (**no**) the Winbind authentication. For example:

```
USEWINBINDAUTH=no
```

**USESMARTCARD=*boolean***

A Boolean to enable (**yes**) or disable (**no**) the smart card authentication. For example:

```
USESMARTCARD=no
```

**USELDAP=*boolean***

A Boolean to enable (**yes**) or disable (**no**) using LDAP for user account configuration. For example:

```
USELDAP=no
```

**USENIS=*boolean***

A Boolean to enable (**yes**) or disable (**no**) using NIS for user account configuration. For example:

```
USENIS=no
```

**USEKERBEROS=*boolean***

A Boolean to enable (**yes**) or disable (**no**) the Kerberos authentication. For example:

```
USEKERBEROS=no
```

**USESYSNETAUTH=*boolean***

A Boolean to enable (**yes**) or disable (**no**) authenticating system accounts with network services. For example:

```
USESYSNETAUTH=no
```

**USESMBAUTH=*boolean***

A Boolean to enable (**yes**) or disable (**no**) the SMB authentication. For example:

```
USESMBAUTH=no
```

**USESSSD=*boolean***

A Boolean to enable (**yes**) or disable (**no**) using SSSD for obtaining user information. For example:

```
USESSSD=no
```

**USEHESIOD=*boolean***

A Boolean to enable (**yes**) or disable (**no**) using the Hesoid name service. For example:

```
USEHESIOD=no
```

See [Chapter 13, Configuring Authentication](#) for more information on this topic.

**D.1.3. /etc/sysconfig/autofs**

The `/etc/sysconfig/autofs` file defines custom options for the automatic mounting of devices. This file controls the operation of the automount daemons, which automatically mount file systems when you use them and unmount them after a period of inactivity. File systems can include network file systems, CD-ROM drives, diskettes, and other media.

By default, it contains the following options:

**MASTER\_MAP\_NAME=*value***

The default name for the master map. For example:

```
MASTER_MAP_NAME="auto.master"
```

**TIMEOUT=*value***

The default mount timeout. For example:

```
TIMEOUT=300
```

**NEGATIVE\_TIMEOUT=*value***

The default negative timeout for unsuccessful mount attempts. For example:

```
NEGATIVE_TIMEOUT=60
```

**MOUNT\_WAIT=*value***

The time to wait for a response from `mount`. For example:

```
MOUNT_WAIT=-1
```

**UMOUNT\_WAIT=*value***

The time to wait for a response from `umount`. For example:

```
UMOUNT_WAIT=12
```

**BROWSE\_MODE=*boolean***

A Boolean to enable (**yes**) or disable (**no**) browsing the maps. For example:

```
BROWSE_MODE="no"
```

**MOUNT\_NFS\_DEFAULT\_PROTOCOL=*value***

The default protocol to be used by `mount.nfs`. For example:

```
MOUNT_NFS_DEFAULT_PROTOCOL=4
```

**APPEND\_OPTIONS=*boolean***

A Boolean to enable (**yes**) or disable (**no**) appending the global options instead of replacing them. For example:

```
APPEND_OPTIONS="yes"
```

**LOGGING=*value***

The default logging level. The *value* has to be either **none**, **verbose**, or **debug**. For example:

```
LOGGING="none"
```

**LDAP\_URI=*value***

A space-separated list of server URIs in the form of **protocol://server**. For example:

```
LDAP_URI="ldaps://ldap.example.com/"
```

**LDAP\_TIMEOUT=*value***

The synchronous API calls timeout. For example:

```
LDAP_TIMEOUT=-1
```

**LDAP\_NETWORK\_TIMEOUT=*value***

The network response timeout. For example:

```
LDAP_NETWORK_TIMEOUT=8
```

**SEARCH\_BASE=*value***

The base Distinguished Name (DN) for the map search. For example:

```
SEARCH_BASE=""
```

**AUTH\_CONF\_FILE=*value***

The default location of the SASL authentication configuration file. For example:

```
AUTH_CONF_FILE="/etc/autofs_ldap_auth.conf"
```

**MAP\_HASH\_TABLE\_SIZE=*value***

The hash table size for the map cache. For example:

```
MAP_HASH_TABLE_SIZE=1024
```

**USE\_MISC\_DEVICE=boolean**

A Boolean to enable (**yes**) or disable (**no**) using the autofs miscellaneous device. For example:

```
USE_MISC_DEVICE="yes"
```

**OPTIONS=value**

Additional options to be passed to the LDAP daemon. For example:

```
OPTIONS=""
```

**D.1.4. /etc/sysconfig/clock**

The `/etc/sysconfig/clock` file controls the interpretation of values read from the system hardware clock. It is used by the **Date/Time Properties** tool, and should not be edited by hand. By default, it contains the following option:

**ZONE=value**

The time zone file under `/usr/share/zoneinfo` that `/etc/localtime` is a copy of. For example:

```
ZONE="Europe/Prague"
```

See [Section 2.1, “Date/Time Properties Tool”](#) for more information on the **Date/Time Properties** tool and its usage.

**D.1.5. /etc/sysconfig/dhcpd**

The `/etc/sysconfig/dhcpd` file is used to pass arguments to the **dhcpd** daemon at boot time. By default, it contains the following options:

**DHCPDARGS=value**

Additional options to be passed to the **dhcpd** daemon. For example:

```
DHCPDARGS=
```

See [Chapter 16, DHCP Servers](#) for more information on DHCP and its usage.

**D.1.6. /etc/sysconfig/firstboot**

The `/etc/sysconfig/firstboot` file defines whether to run the **firstboot** utility. By default, it contains the following option:

**RUN\_FIRSTBOOT=boolean**

A Boolean to enable (**YES**) or disable (**NO**) running the **firstboot** program. For example:

```
RUN_FIRSTBOOT=NO
```

The first time the system boots, the **init** program calls the `/etc/rc.d/init.d/firstboot` script, which looks for the `/etc/sysconfig/firstboot` file. If this file does not contain the **RUN\_FIRSTBOOT=NO** option, the **firstboot** program is run, guiding a user through the initial configuration of the system.



#### NOTE

To start the **firstboot** program the next time the system boots, change the value of **RUN\_FIRSTBOOT** option to **YES**, and type the following at a shell prompt:

```
~]# chkconfig firstboot on
```

### D.1.7. `/etc/sysconfig/i18n`

The `/etc/sysconfig/i18n` configuration file defines the default language, any supported languages, and the default system font. By default, it contains the following options:

#### **LANG=***value*

The default language. For example:

```
LANG="en_US.UTF-8"
```

#### **SUPPORTED=***value*

A colon-separated list of supported languages. For example:

```
SUPPORTED="en_US.UTF-8:en_US:en"
```

#### **SYSFONT=***value*

The default system font. For example:

```
SYSFONT="latarcyrheb-sun16"
```

### D.1.8. `/etc/sysconfig/init`

The `/etc/sysconfig/init` file controls how the system appears and functions during the boot process. By default, it contains the following options:

#### **BOOTUP=***value*

The bootup style. The value has to be either **color** (the standard color boot display), **verbose** (an old style display which provides more information), or anything else for the new style display, but without ANSI formatting. For example:

```
BOOTUP=color
```

#### **RES\_COL=***value*

The number of the column in which the status labels start. For example:

```
RES_COL=60
```

-

**MOVE\_TO\_COL=*value***

The terminal sequence to move the cursor to the column specified in **RES\_COL** (see above). For example:

```
MOVE_TO_COL="echo -en \\033[${RES_COL}G"
```

**SETCOLOR\_SUCCESS=*value***

The terminal sequence to set the success color. For example:

```
SETCOLOR_SUCCESS="echo -en \\033[0;32m"
```

**SETCOLOR\_FAILURE=*value***

The terminal sequence to set the failure color. For example:

```
SETCOLOR_FAILURE="echo -en \\033[0;31m"
```

**SETCOLOR\_WARNING=*value***

The terminal sequence to set the warning color. For example:

```
SETCOLOR_WARNING="echo -en \\033[0;33m"
```

**SETCOLOR\_NORMAL=*value***

The terminal sequence to set the default color. For example:

```
SETCOLOR_NORMAL="echo -en \\033[0;39m"
```

**LOGLEVEL=*value***

The initial console logging level. The *value* has to be in the range from **1** (kernel panics only) to **8** (everything, including the debugging information). For example:

```
LOGLEVEL=3
```

**PROMPT=*boolean***

A Boolean to enable (**yes**) or disable (**no**) the hotkey interactive startup. For example:

```
PROMPT=yes
```

**AUTOSWAP=*boolean***

A Boolean to enable (**yes**) or disable (**no**) probing for devices with swap signatures. For example:

```
AUTOSWAP=no
```

**ACTIVE\_CONSOLES=*value***

The list of active consoles. For example:



```
ACTIVE_CONSOLES=/dev/tty[1-6]
```

**SINGLE=value**

The single-user mode type. The *value* has to be either `/sbin/sulogin` (a user will be prompted for a password to log in), or `/sbin/sushell` (the user will be logged in directly). For example:

```
SINGLE=/sbin/sushell
```

**D.1.9. /etc/sysconfig/ip6tables-config**

The `/etc/sysconfig/ip6tables-config` file stores information used by the kernel to set up IPv6 packet filtering at boot time or whenever the `ip6tables` service is started. Note that you should not modify it unless you are familiar with `ip6tables` rules. By default, it contains the following options:

**IP6TABLES\_MODULES=value**

A space-separated list of helpers to be loaded after the firewall rules are applied. For example:

```
IP6TABLES_MODULES="ip_nat_ftp ip_nat_irc"
```

**IP6TABLES\_MODULES\_UNLOAD=boolean**

A Boolean to enable (**yes**) or disable (**no**) module unloading when the firewall is stopped or restarted. For example:

```
IP6TABLES_MODULES_UNLOAD="yes"
```

**IP6TABLES\_SAVE\_ON\_STOP=boolean**

A Boolean to enable (**yes**) or disable (**no**) saving the current firewall rules when the firewall is stopped. For example:

```
IP6TABLES_SAVE_ON_STOP="no"
```

**IP6TABLES\_SAVE\_ON\_RESTART=boolean**

A Boolean to enable (**yes**) or disable (**no**) saving the current firewall rules when the firewall is restarted. For example:

```
IP6TABLES_SAVE_ON_RESTART="no"
```

**IP6TABLES\_SAVE\_COUNTER=boolean**

A Boolean to enable (**yes**) or disable (**no**) saving the rule and chain counters. For example:

```
IP6TABLES_SAVE_COUNTER="no"
```

**IP6TABLES\_STATUS\_NUMERIC=boolean**

A Boolean to enable (**yes**) or disable (**no**) printing IP addresses and port numbers in a numeric format in the status output. For example:

```
IP6TABLES_STATUS_NUMERIC="yes"
```

#### **IP6TABLES\_STATUS\_VERBOSE=boolean**

A Boolean to enable (**yes**) or disable (**no**) printing information about the number of packets and bytes in the status output. For example:

```
IP6TABLES_STATUS_VERBOSE="no"
```

#### **IP6TABLES\_STATUS\_LINENUMBERS=boolean**

A Boolean to enable (**yes**) or disable (**no**) printing line numbers in the status output. For example:

```
IP6TABLES_STATUS_LINENUMBERS="yes"
```



#### **NOTE**

You can create the rules manually using the **ip6tables** command. Once created, type the following at a shell prompt:

```
~]# service ip6tables save
```

This will add the rules to **/etc/sysconfig/ip6tables**. Once this file exists, any firewall rules saved in it persist through a system reboot or a service restart.

### **D.1.10. /etc/sysconfig/keyboard**

The **/etc/sysconfig/keyboard** file controls the behavior of the keyboard. By default, it contains the following options:

#### **KEYTABLE=value**

The name of a keytable file. The files that can be used as keytables start in the **/lib/kbd/keymaps/i386/** directory, and branch into different keyboard layouts from there, all labeled **value.kmap.gz**. The first file name that matches the **KEYTABLE** setting is used. For example:

```
KEYTABLE="us"
```

#### **MODEL=value**

The keyboard model. For example:

```
MODEL="pc105+inet"
```

#### **LAYOUT=value**

The keyboard layout. For example:

```
LAYOUT="us"
```

#### **KEYBOARDTYPE=value**

The keyboard type. Allowed values are **pc** (a PS/2 keyboard), or **sun** (a Sun keyboard). For example:

```
KEYBOARDTYPE="pc"
```

### D.1.11. /etc/sysconfig/ldap

The `/etc/sysconfig/ldap` file holds the basic configuration for the LDAP server. By default, it contains the following options:

#### **SLAPD\_OPTIONS=value**

Additional options to be passed to the **slapd** daemon. For example:

```
SLAPD_OPTIONS="-4"
```

#### **SLURPD\_OPTIONS=value**

Additional options to be passed to the **slurpd** daemon. For example:

```
SLURPD_OPTIONS=""
```

#### **SLAPD\_LDAP=boolean**

A Boolean to enable (**yes**) or disable (**no**) using the LDAP over TCP (that is, **ldapi:///**). For example:

```
SLAPD_LDAP="yes"
```

#### **SLAPD\_LDAPI=boolean**

A Boolean to enable (**yes**) or disable (**no**) using the LDAP over IPC (that is, **ldapi:///**). For example:

```
SLAPD_LDAPI="no"
```

#### **SLAPD\_LDAPS=boolean**

A Boolean to enable (**yes**) or disable (**no**) using the LDAP over TLS (that is, **ldaps:///**). For example:

```
SLAPD_LDAPS="no"
```

#### **SLAPD\_URLS=value**

A space-separated list of URLs. For example:

```
SLAPD_URLS="ldapi:///var/lib/ldap_root/ldapi ldapi:/// ldaps://"
```

#### **SLAPD\_SHUTDOWN\_TIMEOUT=value**

The time to wait for **slapd** to shut down. For example:

```
SLAPD_SHUTDOWN_TIMEOUT=3
```

### **SLAPD\_ULIMIT\_SETTINGS=*value***

The parameters to be passed to **ulimit** before the **slapd** daemon is started. For example:

```
SLAPD_ULIMIT_SETTINGS=""
```

See [Section 20.1, “OpenLDAP”](#) for more information on LDAP and its configuration.

## **D.1.12. /etc/sysconfig/named**

The **/etc/sysconfig/named** file is used to pass arguments to the **named** daemon at boot time. By default, it contains the following options:

### **ROOTDIR=*value***

The chroot environment under which the **named** daemon runs. The *value* has to be a full directory path. For example:

```
ROOTDIR="/var/named/chroot"
```

Note that the chroot environment has to be configured first (type **info chroot** at a shell prompt for more information).

### **OPTIONS=*value***

Additional options to be passed to **named**. For example:

```
OPTIONS="-6"
```

Note that you should not use the **-t** option. Instead, use **ROOTDIR** as described above.

### **KEYTAB\_FILE=*value***

The keytab file name. For example:

```
KEYTAB_FILE="/etc/named.keytab"
```

See [Section 17.2, “BIND”](#) for more information on the BIND DNS server and its configuration.

## **D.1.13. /etc/sysconfig/network**

The **/etc/sysconfig/network** file is used to specify information about the desired network configuration. By default, it contains the following options:

### **NETWORKING=*boolean***

A Boolean to enable (**yes**) or disable (**no**) networking. For example:

```
NETWORKING=yes
```

**HOSTNAME=***value*

The host name of the machine. For example:

```
HOSTNAME=penguin.example.com
```

The file may also contain some of the following options:

**GATEWAY=***value*

The IP address of the network's gateway. For example:

```
GATEWAY=192.168.1.1
```

This is used as the default gateway when there is no **GATEWAY** directive in an interface's **ifcfg** file.

**NM\_BOND\_VLAN\_ENABLED=***boolean*

A Boolean to allow (**yes**) or disallow (**no**) the **NetworkManager** application from detecting and managing bonding, bridging, and VLAN interfaces. For example:

```
NM_BOND_VLAN_ENABLED=yes
```

The **NM\_CONTROLLED** directive is dependent on this option.

**NOTE**

If you want to completely disable IPv6, you should add these lines to `/etc/sysctl.conf`:

```
net.ipv6.conf.all.disable_ipv6=1
```

```
net.ipv6.conf.default.disable_ipv6=1
```

In addition, adding **ipv6.disable=1** to the kernel command line will disable the kernel module **net-pf-10** which implements IPv6.

**WARNING**

Do not use custom init scripts to configure network settings. When performing a post-boot network service restart, custom init scripts configuring network settings that are run outside of the network init script lead to unpredictable results.

**D.1.14. /etc/sysconfig/ntpd**

The `/etc/sysconfig/ntpd` file is used to pass arguments to the **ntpd** daemon at boot time. By default, it contains the following option:

**OPTIONS=*value***

Additional options to be passed to **ntpd**. For example:

```
OPTIONS="-u ntp:ntp -p /var/run/ntpd.pid -g"
```

See [Section 2.1.2, “Network Time Protocol Properties”](#) or [Section 2.2.2, “Network Time Protocol Setup”](#) for more information on how to configure the **ntpd** daemon.

**D.1.15. /etc/sysconfig/quagga**

The **/etc/sysconfig/quagga** file holds the basic configuration for Quagga daemons. By default, it contains the following options:

**QCONFDIR=*value***

The directory with the configuration files for Quagga daemons. For example:

```
QCONFDIR="/etc/quagga"
```

**BGPD\_OPTS=*value***

Additional options to be passed to the **bgpd** daemon. For example:

```
BGPD_OPTS="-A 127.0.0.1 -f ${QCONFDIR}/bgpd.conf"
```

**OSPF6D\_OPTS=*value***

Additional options to be passed to the **ospf6d** daemon. For example:

```
OSPF6D_OPTS="-A ::1 -f ${QCONFDIR}/ospf6d.conf"
```

**OSPFD\_OPTS=*value***

Additional options to be passed to the **ospfd** daemon. For example:

```
OSPFD_OPTS="-A 127.0.0.1 -f ${QCONFDIR}/ospfd.conf"
```

**RIPD\_OPTS=*value***

Additional options to be passed to the **ripd** daemon. For example:

```
RIPD_OPTS="-A 127.0.0.1 -f ${QCONFDIR}/ripd.conf"
```

**RIPNGD\_OPTS=*value***

Additional options to be passed to the **ripngd** daemon. For example:

```
RIPNGD_OPTS="-A ::1 -f ${QCONFDIR}/ripngd.conf"
```

**ZEBRA\_OPTS=*value***

Additional options to be passed to the **zebra** daemon. For example:

■

```
ZEBRA_OPTS="-A 127.0.0.1 -f ${QCONFDIR}/zebra.conf"
```

**ISISD\_OPTS=value**

Additional options to be passed to the **isisd** daemon. For example:

```
ISISD_OPTS="-A :::1 -f ${QCONFDIR}/isisd.conf"
```

**WATCH\_OPTS=value**

Additional options to be passed to the **watchquagga** daemon. For example:

```
WATCH_OPTS="-Az -b_ -r/sbin/service_%s_restart -s/sbin/service_%s_start  
-k/sbin/service_%s_stop"
```

**WATCH\_DAEMONS=value**

A space separated list of monitored daemons. For example:

```
WATCH_DAEMONS="zebra bgpd ospfd ospf6d ripd ripngd"
```

### D.1.16. /etc/sysconfig/radvd

The **/etc/sysconfig/radvd** file is used to pass arguments to the **radvd** daemon at boot time. By default, it contains the following option:

**OPTIONS=value**

Additional options to be passed to the **radvd** daemon. For example:

```
OPTIONS="-u radvd"
```

### D.1.17. /etc/sysconfig/samba

The **/etc/sysconfig/samba** file is used to pass arguments to the Samba daemons at boot time. By default, it contains the following options:

**SMBDOPTIONS=value**

Additional options to be passed to **smbd**. For example:

```
SMBDOPTIONS="-D"
```

**NMBDOPTIONS=value**

Additional options to be passed to **nmbd**. For example:

```
NMBDOPTIONS="-D"
```

**WINBINDOPTIONS=value**

Additional options to be passed to **winbindd**. For example:

-

```
WINBINDOPTIONS=""
```

See [Section 21.1, “Samba”](#) for more information on Samba and its configuration.

### D.1.18. `/etc/sysconfig/saslauthd`

The `/etc/sysconfig/saslauthd` file is used to control which arguments are passed to `saslauthd`, the SASL authentication server. By default, it contains the following options:

#### **SOCKETDIR=***value*

The directory for the `saslauthd`'s listening socket. For example:

```
SOCKETDIR=/var/run/saslauthd
```

#### **MECH=***value*

The authentication mechanism to use to verify user passwords. For example:

```
MECH=pam
```

#### **DAEMONOPTS=***value*

Options to be passed to the `daemon()` function that is used by the `/etc/rc.d/init.d/saslauthd` init script to start the `saslauthd` service. For example:

```
DAEMONOPTS="--user saslauthd"
```

#### **FLAGS=***value*

Additional options to be passed to the `saslauthd` service. For example:

```
FLAGS=
```

### D.1.19. `/etc/sysconfig/selinux`

The `/etc/sysconfig/selinux` file contains the basic configuration options for SELinux. It is a symbolic link to `/etc/selinux/config`, and by default, it contains the following options:

#### **SELINUX=***value*

The security policy. The *value* can be either **enforcing** (the security policy is always enforced), **permissive** (instead of enforcing the policy, appropriate warnings are displayed), or **disabled** (no policy is used). For example:

```
SELINUX=enforcing
```

#### **SELINUXTYPE=***value*

The protection type. The *value* can be either **targeted** (the targeted processes are protected), or **m1s** (the Multi Level Security protection). For example:

```
SELINUXTYPE=targeted
```



■

### D.1.20. /etc/sysconfig/sendmail

The `/etc/sysconfig/sendmail` is used to set the default values for the **Sendmail** application. By default, it contains the following values:

#### **DAEMON=boolean**

A Boolean to enable (**yes**) or disable (**no**) running **sendmail** as a daemon. For example:

```
DAEMON=yes
```

#### **QUEUE=value**

The interval at which the messages are to be processed. For example:

```
QUEUE=1h
```

See [Section 19.3.2, “Sendmail”](#) for more information on Sendmail and its configuration.

### D.1.21. /etc/sysconfig/spamassassin

The `/etc/sysconfig/spamassassin` file is used to pass arguments to the **spamd** daemon (a daemonized version of **Spamassassin**) at boot time. By default, it contains the following option:

#### **SPAMDOPTIONS=value**

Additional options to be passed to the **spamd** daemon. For example:

```
SPAMDOPTIONS="-d -c -m5 -H"
```

See [Section 19.4.2.6, “Spam Filters”](#) for more information on Spamassassin and its configuration.

### D.1.22. /etc/sysconfig/squid

The `/etc/sysconfig/squid` file is used to pass arguments to the **squid** daemon at boot time. By default, it contains the following options:

#### **SQUID\_OPTS=value**

Additional options to be passed to the **squid** daemon. For example:

```
SQUID_OPTS=""
```

#### **SQUID\_SHUTDOWN\_TIMEOUT=value**

The time to wait for **squid** daemon to shut down. For example:

```
SQUID_SHUTDOWN_TIMEOUT=100
```

#### **SQUID\_CONF=value**

The default configuration file. For example:

```
SQUID_CONF="/etc/squid/squid.conf"
```

### D.1.23. /etc/sysconfig/system-config-users

The `/etc/sysconfig/system-config-users` file is the configuration file for the **User Manager** utility, and should not be edited by hand. By default, it contains the following options:

#### **FILTER=boolean**

A Boolean to enable (**true**) or disable (**false**) filtering of system users. For example:

```
FILTER=true
```

#### **ASSIGN\_HIGHEST\_UID=boolean**

A Boolean to enable (**true**) or disable (**false**) assigning the highest available UID to newly added users. For example:

```
ASSIGN_HIGHEST_UID=true
```

#### **ASSIGN\_HIGHEST\_GID=boolean**

A Boolean to enable (**true**) or disable (**false**) assigning the highest available GID to newly added groups. For example:

```
ASSIGN_HIGHEST_GID=true
```

#### **PREFER\_SAME\_UID\_GID=boolean**

A Boolean to enable (**true**) or disable (**false**) using the same UID and GID for newly added users when possible. For example:

```
PREFER_SAME_UID_GID=true
```

See [Section 3.2, “Managing Users via the User Manager Application”](#) for more information on **User Manager** and its usage.

### D.1.24. /etc/sysconfig/vncservers

The `/etc/sysconfig/vncservers` file configures the way the *Virtual Network Computing (VNC)* server starts up. By default, it contains the following options:

#### **VNCSERVERS=value**

A list of space separated **display:username** pairs. For example:

```
VNCSERVERS="2:myusername"
```

#### **VNCSERVERARGS[display]=value**

Additional arguments to be passed to the VNC server running on the specified *display*. For example:

```
VNCSERVERARGS[2]="-geometry 800x600 -nolisten tcp -localhost"
```

### D.1.25. /etc/sysconfig/xinetd

The `/etc/sysconfig/xinetd` file is used to pass arguments to the `xinetd` daemon at boot time. By default, it contains the following options:

#### **EXTRAOPTIONS=***value*

Additional options to be passed to `xinetd`. For example:

```
EXTRAOPTIONS=""
```

#### **XINETD\_LANG=***value*

The locale information to be passed to every service started by `xinetd`. Note that to remove locale information from the `xinetd` environment, you can use an empty string (""), or `none`. For example:

```
XINETD_LANG="en_US"
```

See [Chapter 12, Services and Daemons](#) for more information on how to configure the `xinetd` services.

## D.2. DIRECTORIES IN THE /ETC/SYSCONFIG/ DIRECTORY

The following directories are normally found in `/etc/sysconfig/`.

### **/etc/sysconfig/cbq/**

This directory contains the configuration files needed to do *Class Based Queuing* for bandwidth management on network interfaces. CBQ divides user traffic into a hierarchy of classes based on any combination of IP addresses, protocols, and application types.

### **/etc/sysconfig/networking/**

This directory is used by the now deprecated **Network Administration Tool (system-config-network)**, and its contents should not be edited manually. For more information about configuring network interfaces using graphical configuration tools, see [Chapter 10, NetworkManager](#).

### **/etc/sysconfig/network-scripts/**

This directory contains the following network-related configuration files:

- Network configuration files for each configured network interface, such as `ifcfg-eth0` for the `eth0` Ethernet interface.
- Scripts used to bring network interfaces up and down, such as `ifup` and `ifdown`.
- Scripts used to bring ISDN interfaces up and down, such as `ifup-isdn` and `ifdown-isdn`.
- Various shared network function scripts which should not be edited directly.

For more information on the `/etc/sysconfig/network-scripts/` directory, see [Chapter 11, Network Interfaces](#).

### **`/etc/sysconfig/rhn/`**

This directory contains the configuration files and GPG keys for Red Hat Network. No files in this directory should be edited by hand. For more information on Red Hat Network, see the Red Hat Network website online at <https://rhn.redhat.com/>.

## **D.3. ADDITIONAL RESOURCES**

This chapter is only intended as an introduction to the files in the `/etc/sysconfig/` directory. The following source contains more comprehensive information.

### **D.3.1. Installed Documentation**

#### **`/usr/share/doc/ini-scripts-version/sysconfig.txt`**

A more authoritative listing of the files found in the `/etc/sysconfig/` directory and the configuration options available for them.

## APPENDIX E. THE PROC FILE SYSTEM

The Linux kernel has two primary functions: to control access to physical devices on the computer and to schedule when and how processes interact with these devices. The `/proc/` directory (also called the **proc** file system) contains a hierarchy of special files which represent the current state of the kernel, allowing applications and users to peer into the kernel's view of the system.

The `/proc/` directory contains a wealth of information detailing system hardware and any running processes. In addition, some of the files within `/proc/` can be manipulated by users and applications to communicate configuration changes to the kernel.



### NOTE

Later versions of the 2.6 kernel have made the `/proc/ide/` and `/proc/pci/` directories obsolete. The `/proc/ide/` file system is now superseded by files in **sysfs**; to retrieve information on PCI devices, use **lspci** instead. For more information on **sysfs** or **lspci**, see their respective **man** pages.

### E.1. A VIRTUAL FILE SYSTEM

Linux systems store all data as *files*. Most users are familiar with the two primary types of files: text and binary. But the `/proc/` directory contains another type of file called a *virtual file*. As such, `/proc/` is often referred to as a *virtual file system*.

Virtual files have unique qualities. Most of them are listed as zero bytes in size, but can still contain a large amount of information when viewed. In addition, most of the time and date stamps on virtual files reflect the current time and date, indicative of the fact they are constantly updated.

Virtual files such as `/proc/interrupts`, `/proc/meminfo`, `/proc/mounts`, and `/proc/partitions` provide an up-to-the-moment glimpse of the system's hardware. Others, like the `/proc/filesystems` file and the `/proc/sys/` directory provide system configuration information and interfaces.

For organizational purposes, files containing information on a similar topic are grouped into virtual directories and sub-directories. Process directories contain information about each running process on the system.

#### E.1.1. Viewing Virtual Files

Most files within `/proc/` files operate similarly to text files, storing useful system and hardware data in human-readable text format. As such, you can use **cat**, **more**, or **less** to view them. For example, to display information about the system's CPU, run **cat /proc/cpuinfo**. This will return output similar to the following:

```
processor : 0
vendor_id : AuthenticAMD
cpu family : 5
model    : 9
model name : AMD-K6(tm) 3D+
Processor stepping : 1 cpu
MHz      : 400.919
cache size : 256 KB
fdiv_bug : no
```

```

hlt_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 1
wp : yes
flags : fpu vme de pse tsc msr mce cx8 pge mmx syscall 3dnow k6_mtrr
bogomips : 799.53

```

Some files in **/proc/** contain information that is not human-readable. To retrieve information from such files, use tools such as **lspci**, **apm**, **free**, and **top**.



## NOTE

Some of the virtual files in the **/proc/** directory are readable only by the root user.

### E.1.2. Changing Virtual Files

As a general rule, most virtual files within the **/proc/** directory are read-only. However, some can be used to adjust settings in the kernel. This is especially true for files in the **/proc/sys/** subdirectory.

To change the value of a virtual file, use the following command:

```
echo value > /proc/file
```

For example, to change the host name on the fly, run:

```
echo www.example.com > /proc/sys/kernel/hostname
```

Other files act as binary or Boolean switches. Typing **cat /proc/sys/net/ipv4/ip\_forward** returns either a **0** (off or false) or a **1** (on or true). A **0** indicates that the kernel is not forwarding network packets. To turn packet forwarding on, run **echo 1 > /proc/sys/net/ipv4/ip\_forward**.



## NOTE

Another command used to alter settings in the **/proc/sys/** subdirectory is **/sbin/sysctl**. For more information on this command, see [Section E.4, “Using the sysctl Command”](#)

For a listing of some of the kernel configuration files available in the **/proc/sys/** subdirectory, see [Section E.3.9, “/proc/sys/”](#).

## E.2. TOP-LEVEL FILES WITHIN THE `PROC` FILE SYSTEM

Below is a list of some of the more useful virtual files in the top-level of the **/proc/** directory.



## NOTE

In most cases, the content of the files listed in this section are not the same as those installed on your machine. This is because much of the information is specific to the hardware on which Red Hat Enterprise Linux is running for this documentation effort.

### E.2.1. /proc/buddyinfo

The `/proc/buddyinfo` file is used primarily for diagnosing memory fragmentation issues. The output depends on the memory layout used, which is architecture specific. The following is an example from a 32-bit system:

```
Node 0, zone    DMA      90      6      2      1      1      ...
Node 0, zone   Normal  1650    310    5      0      0      ...
Node 0, zone   HighMem   2       0      0      1      1      ...
```

Using the buddy algorithm, each column represents the number of memory pages of a certain order, a certain size, that are available at any given time. In the example above, for zone **DMA**, there are 90 of  $2^0 \times \text{PAGE\_SIZE}$  bytes large chunks of memory. Similarly, there are 6 of  $2^1 \times \text{PAGE\_SIZE}$  chunks and 2 of  $2^2 \times \text{PAGE\_SIZE}$  chunks of memory available.

The **DMA** row references the first 16 MB of memory on the system, the **HighMem** row references all memory greater than 896 MB on the system, and the **Normal** row references the memory in between.

On a 64-bit system, the output might look as follows:

```
Node 0, zone    DMA      0      3      1      2      4      3      1      2      3
3  1
Node 0, zone   DMA32    295    25850   7065   1645   835   220   78   6   0
1  0
Node 0, zone   Normal   3824    3359    736    159    31    3    1    1    1
1  0
```

The **DMA** row references the first 16 MB of memory on the system, the **DMA32** row references all memory allocated for devices that cannot address memory greater than 4 GB, and the **Normal** row references all memory above the **DMA32** allocation, which includes all memory above 4 GB on the system.

### E.2.2. /proc/cmdline

This file shows the parameters passed to the kernel at the time it is started. A sample `/proc/cmdline` file looks like the following:

```
ro root=/dev/VolGroup00/LogVol100 rhgb quiet 3
```

This tells us that the kernel is mounted read-only (signified by **(ro)**), located on the first logical volume (**LogVol100**) of the first volume group (**/dev/VolGroup00**). **LogVol100** is the equivalent of a disk partition in a non-LVM system (Logical Volume Management), just as **/dev/VolGroup00** is similar in concept to **/dev/hda1**, but much more extensible.

For more information on LVM used in Red Hat Enterprise Linux, see <http://www.tldp.org/HOWTO/LVM-HOWTO/index.html>.

Next, **rhgb** signals that the **rhgb** package has been installed, and graphical booting is supported, assuming `/etc/inittab` shows a default runlevel set to **id:5:initdefault:**.

Finally, **quiet** indicates all verbose kernel messages are suppressed at boot time.

### E.2.3. /proc/cpuinfo

This virtual file identifies the type of processor used by your system. The following is an example of the output typical of `/proc/cpuinfo`:

```
processor : 0
vendor_id : GenuineIntel
cpu family : 15
model    : 2
model name : Intel(R) Xeon(TM) CPU 2.40GHz
stepping : 7 cpu
MHz      : 2392.371
cache size : 512 KB
physical id : 0
siblings : 2
runqueue : 0
fdiv_bug : no
hlt_bug  : no
f00f_bug : no
coma_bug : no
fpu      : yes
fpu_exception : yes
cpuid level : 2
wp       : yes
flags    : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm
bogomips : 4771.02
```

- **processor** — Provides each processor with an identifying number. On systems that have one processor, only a `0` is present.
- **cpu family** — Authoritatively identifies the type of processor in the system. For an Intel-based system, place the number in front of "86" to determine the value. This is particularly helpful for those attempting to identify the architecture of an older system such as a 586, 486, or 386. Because some RPM packages are compiled for each of these particular architectures, this value also helps users determine which packages to install.
- **model name** — Displays the common name of the processor, including its project name.
- **cpu MHz** — Shows the precise speed in megahertz for the processor to the thousandths decimal place.
- **cache size** — Displays the amount of level 2 memory cache available to the processor.
- **siblings** — Displays the total number of sibling CPUs on the same physical CPU for architectures which use hyper-threading.
- **flags** — Defines a number of different qualities about the processor, such as the presence of a floating point unit (FPU) and the ability to process MMX instructions.

#### E.2.4. `/proc/crypto`

This file lists all installed cryptographic ciphers used by the Linux kernel, including additional details for each. A sample `/proc/crypto` file looks like the following:

```
name          : sha1
module       : kernel
```



```
type      : digest
blocksize : 64
digestsize : 20
name      : md5
module    : md5
type      : digest
blocksize : 64
digestsize : 16
```

### E.2.5. `/proc/devices`

This file displays the various character and block devices currently configured (not including devices whose modules are not loaded). Below is a sample output from this file:

```
Character devices:
 1 mem
 4 /dev/vc/0
 4 tty
 4 ttyS
 5 /dev/tty
 5 /dev/console
 5 /dev/ptmx
 7 vcs
10 misc
13 input
29 fb
36 netlink
128 ptm
136 pts
180 usb

Block devices:
 1 ramdisk
 3 ide0
 9 md
22 ide1
253 device-mapper
254 mdp
```

The output from `/proc/devices` includes the major number and name of the device, and is broken into two major sections: **Character devices** and **Block devices**.

*Character devices* are similar to *block devices*, except for two basic differences:

1. Character devices do not require buffering. Block devices have a buffer available, allowing them to order requests before addressing them. This is important for devices designed to store information — such as hard drives — because the ability to order the information before writing it to the device allows it to be placed in a more efficient order.
2. Character devices send data with no preconfigured size. Block devices can send and receive information in blocks of a size configured per device.

For more information about devices, see the `devices.txt` file in the kernel-doc package (see [Section E.5, “Additional Resources”](#)).

## E.2.6. /proc/dma

This file contains a list of the registered ISA DMA channels in use. A sample `/proc/dma` file looks like the following:

```
4: cascade
```

## E.2.7. /proc/execdomains

This file lists the *execution domains* currently supported by the Linux kernel, along with the range of personalities they support.

```
0-0 Linux [kernel]
```

Think of execution domains as the "personality" for an operating system. Because other binary formats, such as Solaris, UnixWare, and FreeBSD, can be used with Linux, programmers can change the way the operating system treats system calls from these binaries by changing the personality of the task. Except for the **PER\_LINUX** execution domain, different personalities can be implemented as dynamically loadable modules.

## E.2.8. /proc/fb

This file contains a list of frame buffer devices, with the frame buffer device number and the driver that controls it. Typical output of `/proc/fb` for systems which contain frame buffer devices looks similar to the following:

```
0 VESA VGA
```

## E.2.9. /proc/filesystems

This file displays a list of the file system types currently supported by the kernel. Sample output from a generic `/proc/filesystems` file looks similar to the following:

```
nodev sysfs
nodev rootfs
nodev bdev
nodev proc
nodev sockfs
nodev binfmt_misc
nodev usbfs
nodev usbdevfs
nodev futexfs
nodev tmpfs
nodev pipefs
nodev eventpollfs
nodev devpts
  ext2
nodev ramfs
nodev hugetlbfs
  iso9660
nodev mqueue
```

```

ext3
nodev   rpc_pipefs
nodev   autofs

```

The first column signifies whether the file system is mounted on a block device. Those beginning with **nodev** are not mounted on a device. The second column lists the names of the file systems supported.

The **mount** command cycles through the file systems listed here when one is not specified as an argument.

## E.2.10. /proc/interrupts

This file records the number of interrupts per IRQ on the x86 architecture. A standard **/proc/interrupts** looks similar to the following:

```

CPU0
0:   80448940          XT-PIC  timer
1:   174412           XT-PIC  keyboard
2:     0              XT-PIC  cascade
8:     1              XT-PIC  rtc
10:  410964           XT-PIC  eth0
12:  60330            XT-PIC  PS/2 Mouse
14:  1314121          XT-PIC  ide0
15:  5195422          XT-PIC  ide1
NMI:     0
ERR:     0

```

For a multi-processor machine, this file may look slightly different:

```

      CPU0          CPU1
0: 1366814704      0          XT-PIC  timer
1:   128          340       IO-APIC-edge  keyboard
2:     0           0          XT-PIC  cascade
8:     0           1       IO-APIC-edge  rtc
12:   5323        5793       IO-APIC-edge  PS/2 Mouse
13:     1           0          XT-PIC  fpu
16: 11184294     15940594     IO-APIC-level  Intel EtherExpress Pro 10/100
Ethernet
20:   8450043     11120093     IO-APIC-level  megaraid
30:   10432        10722     IO-APIC-level  aic7xxx
31:     23         22     IO-APIC-level  aic7xxx
NMI:     0
ERR:     0

```

The first column refers to the IRQ number. Each CPU in the system has its own column and its own number of interrupts per IRQ. The next column reports the type of interrupt, and the last column contains the name of the device that is located at that IRQ.

Each of the types of interrupts seen in this file, which are architecture-specific, mean something different. For x86 machines, the following values are common:

- **XT-PIC** — This is the old AT computer interrupts.
- **IO-APIC-edge** — The voltage signal on this interrupt transitions from low to high, creating an *edge*, where the interrupt occurs and is only signaled once. This kind of interrupt, as well as the

**IO-APIC-level** interrupt, are only seen on systems with processors from the 586 family and higher.

- **IO-APIC-level** — Generates interrupts when its voltage signal is high until the signal is low again.

### E.2.11. /proc/iomem

This file shows you the current map of the system's memory for each physical device:

```
00000000-0009fbff : System RAM
0009fc00-0009ffff : reserved
000a0000-000bffff : Video RAM area
000c0000-000c7fff : Video ROM
000f0000-000fffff : System ROM
00100000-07ffffff : System RAM
00100000-00291ba8 : Kernel code
00291ba9-002e09cb : Kernel data
e0000000-e3ffffff : VIA Technologies, Inc. VT82C597 [Apollo VP3] e4000000-
e7ffffff : PCI Bus #01
e4000000-e4003fff : Matrox Graphics, Inc. MGA G200 AGP
e5000000-e57ffffff : Matrox Graphics, Inc. MGA G200 AGP
e8000000-e8ffffff : PCI Bus #01
e8000000-e8ffffff : Matrox Graphics, Inc. MGA G200 AGP
ea000000-ea00007f : Digital Equipment Corporation DECchip 21140
[FasterNet]
ea000000-ea00007f : tulip ffff0000-ffffffff : reserved
```

The first column displays the memory registers used by each of the different types of memory. The second column lists the kind of memory located within those registers and displays which memory registers are used by the kernel within the system RAM or, if the network interface card has multiple Ethernet ports, the memory registers assigned for each port.

### E.2.12. /proc/ioports

The output of **/proc/ioports** provides a list of currently registered port regions used for input or output communication with a device. This file can be quite long. The following is a partial listing:

```
0000-001f : dma1
0020-003f : pic1
0040-005f : timer
0060-006f : keyboard
0070-007f : rtc
0080-008f : dma page reg
00a0-00bf : pic2
00c0-00df : dma2
00f0-00ff : fpu
0170-0177 : ide1
01f0-01f7 : ide0
02f8-02ff : serial(auto)
0376-0376 : ide1
03c0-03df : vga+
03f6-03f6 : ide0
03f8-03ff : serial(auto)
0cf8-0cff : PCI conf1
```

```
d000-dfff : PCI Bus #01
e000-e00f : VIA Technologies, Inc. Bus Master IDE
e000-e007 : ide0
e008-e00f : ide1
e800-e87f : Digital Equipment Corporation DECchip 21140 [FasterNet]
e800-e87f : tulip
```

The first column gives the I/O port address range reserved for the device listed in the second column.

### E.2.13. /proc/kcore

This file represents the physical memory of the system and is stored in the core file format. Unlike most `/proc/` files, `kcore` displays a size. This value is given in bytes and is equal to the size of the physical memory (RAM) used plus 4 KB.

The contents of this file are designed to be examined by a debugger, such as `gdb`, and is not human readable.



#### WARNING

Do not view the `/proc/kcore` virtual file. The contents of the file scramble text output on the terminal. If this file is accidentally viewed, press `Ctrl+C` to stop the process and then type `reset` to bring back the command line prompt.

### E.2.14. /proc/kmsg

This file is used to hold messages generated by the kernel. These messages are then picked up by other programs, such as `/sbin/klogd` or `/bin/dmesg`.

### E.2.15. /proc/loadavg

This file provides a look at the load average in regard to both the CPU and IO over time, as well as additional data used by `uptime` and other commands. A sample `/proc/loadavg` file looks similar to the following:

```
0.20 0.18 0.12 1/80 11206
```

The first three columns measure CPU and IO utilization of the last one, five, and 15 minute periods. The fourth column shows the number of currently running processes and the total number of processes. The last column displays the last process ID used.

In addition, load average also refers to the number of processes ready to run (i.e. in the run queue, waiting for a CPU share).

### E.2.16. /proc/locks

This file displays the files currently locked by the kernel. The contents of this file contain internal kernel debugging data and can vary tremendously, depending on the use of the system. A sample `/proc/locks` file for a lightly loaded system looks similar to the following:

```

1: POSIX  ADVISORY  WRITE 3568 fd:00:2531452 0 EOF
2: FLOCK  ADVISORY  WRITE 3517 fd:00:2531448 0 EOF
3: POSIX  ADVISORY  WRITE 3452 fd:00:2531442 0 EOF
4: POSIX  ADVISORY  WRITE 3443 fd:00:2531440 0 EOF
5: POSIX  ADVISORY  WRITE 3326 fd:00:2531430 0 EOF
6: POSIX  ADVISORY  WRITE 3175 fd:00:2531425 0 EOF
7: POSIX  ADVISORY  WRITE 3056 fd:00:2548663 0 EOF

```

Each lock has its own line which starts with a unique number. The second column refers to the class of lock used, with **FLOCK** signifying the older-style UNIX file locks from a **flock** system call and **POSIX** representing the newer POSIX locks from the **lockf** system call.

The third column can have two values: **ADVISORY** or **MANDATORY**. **ADVISORY** means that the lock does not prevent other people from accessing the data; it only prevents other attempts to lock it. **MANDATORY** means that no other access to the data is permitted while the lock is held. The fourth column reveals whether the lock is allowing the holder **READ** or **WRITE** access to the file. The fifth column shows the ID of the process holding the lock. The sixth column shows the ID of the file being locked, in the format of **MAJOR-DEVICE:MINOR-DEVICE:INODE-NUMBER**. The seventh and eighth column shows the start and end of the file's locked region.

### E.2.17. /proc/mdstat

This file contains the current information for multiple-disk, RAID configurations. If the system does not contain such a configuration, then **/proc/mdstat** looks similar to the following:

```
Personalities : read_ahead not set unused devices: <none>
```

This file remains in the same state as seen above unless a software RAID or **md** device is present. In that case, view **/proc/mdstat** to find the current status of **mdX** RAID devices.

The **/proc/mdstat** file below shows a system with its **md0** configured as a RAID 1 device, while it is currently re-syncing the disks:

```

Personalities : [linear] [raid1] read_ahead 1024 sectors
md0: active raid1 sda2[1] sdb2[0] 9940 blocks [2/2] [UU] resync=1%
finish=12.3min algorithm 2 [3/3] [UUU]
unused devices: <none>

```

### E.2.18. /proc/meminfo

This is one of the more commonly used files in the **/proc/** directory, as it reports a large amount of valuable information about the system's RAM usage.

The following sample **/proc/meminfo** virtual file is from a system with 2 GB of RAM and 1 GB of swap space:

```

MemTotal:      1921988 kB
MemFree:       1374408 kB
Buffers:       32688 kB
Cached:        370540 kB
SwapCached:    0 kB
Active:        344604 kB
Inactive:      80800 kB

```

```

Active(anon):      22364 kB
Inactive(anon):    4 kB
Active(file):      322240 kB
Inactive(file):    80796 kB
Unevictable:      0 kB
Mlocked:          0 kB
SwapTotal:        1048572 kB
SwapFree:         1048572 kB
Dirty:            48 kB
Writeback:        0 kB
AnonPages:        22260 kB
Mapped:           13628 kB
Shmem:            196 kB
Slab:             91648 kB
SReclaimable:     34024 kB
SUnreclaim:       57624 kB
KernelStack:      2880 kB
PageTables:       3620 kB
NFS_Unstable:     0 kB
Bounce:           0 kB
WritebackTmp:     0 kB
CommitLimit:      2009564 kB
Committed_AS:     134216 kB
VmallocTotal:     34359738367 kB
VmallocUsed:      12276 kB
VmallocChunk:     34359712840 kB
HardwareCorrupted: 0 kB
AnonHugePages:    0 kB
HugePages_Total:  0
HugePages_Free:   0
HugePages_Rsvd:   0
HugePages_Surp:   0
Hugepagesize:     2048 kB
DirectMap4k:      8064 kB
DirectMap2M:      2088960 kB

```

While the file shows kilobytes (kB; 1 kB equals 1000 B), it is actually kibibytes (KiB; 1 KiB equals 1024 B). This imprecision in `/proc/meminfo` is known, but is not corrected due to legacy concerns - programs rely on `/proc/meminfo` to specify size with the "kB" string.

Much of the information in `/proc/meminfo` is used by the `free`, `top`, and `ps` commands. In fact, the output of the `free` command is similar in appearance to the contents and structure of `/proc/meminfo`. However, `/proc/meminfo` itself has more details:

- **MemTotal** — Total amount of usable RAM, in kibibytes, which is physical RAM minus a number of reserved bits and the kernel binary code.
- **MemFree** — The amount of physical RAM, in kibibytes, left unused by the system.
- **Buffers** — The amount, in kibibytes, of temporary storage for raw disk blocks.
- **Cached** — The amount of physical RAM, in kibibytes, used as cache memory.
- **SwapCached** — The amount of memory, in kibibytes, that has once been moved into swap, then back into the main memory, but still also remains in the swapfile. This saves I/O, because the memory does not need to be moved into swap again.

- **Active** — The amount of memory, in kibibytes, that has been used more recently and is usually not reclaimed unless absolutely necessary.
- **Inactive** — The amount of memory, in kibibytes, that has been used less recently and is more eligible to be reclaimed for other purposes.
- **Active(anon)** — The amount of anonymous and tmpfs/shmem memory, in kibibytes, that is in active use, or was in active use since the last time the system moved something to swap.
- **Inactive(anon)** — The amount of anonymous and tmpfs/shmem memory, in kibibytes, that is a candidate for eviction.
- **Active(file)** — The amount of file cache memory, in kibibytes, that is in active use, or was in active use since the last time the system reclaimed memory.
- **Inactive(file)** — The amount of file cache memory, in kibibytes, that is newly loaded from the disk, or is a candidate for reclaiming.
- **Unevictable** — The amount of memory, in kibibytes, discovered by the pageout code, that is not evictable because it is locked into memory by user programs.
- **Mlocked** — The total amount of memory, in kibibytes, that is not evictable because it is locked into memory by user programs.
- **SwapTotal** — The total amount of swap available, in kibibytes.
- **SwapFree** — The total amount of swap free, in kibibytes.
- **Dirty** — The total amount of memory, in kibibytes, waiting to be written back to the disk.
- **Writeback** — The total amount of memory, in kibibytes, actively being written back to the disk.
- **AnonPages** — The total amount of memory, in kibibytes, used by pages that are not backed by files and are mapped into userspace page tables.
- **Mapped** — The memory, in kibibytes, used for files that have been mmaped, such as libraries.
- **Shmem** — The total amount of memory, in kibibytes, used by shared memory (shmem) and tmpfs.
- **Slab** — The total amount of memory, in kibibytes, used by the kernel to cache data structures for its own use.
- **SReclaimable** — The part of Slab that can be reclaimed, such as caches.
- **SUnreclaim** — The part of Slab that cannot be reclaimed even when lacking memory.
- **KernelStack** — The amount of memory, in kibibytes, used by the kernel stack allocations done for each task in the system.
- **PageTables** — The total amount of memory, in kibibytes, dedicated to the lowest page table level.
- **NFS\_Unstable** — The amount, in kibibytes, of NFS pages sent to the server but not yet committed to the stable storage.



- **Bounce** — The amount of memory, in kibibytes, used for the block device "bounce buffers".
- **WritebackTmp** — The amount of memory, in kibibytes, used by FUSE for temporary writeback buffers.
- **CommitLimit** — The total amount of memory currently available to be allocated on the system based on the overcommit ratio (`vm.overcommit_ratio`). This limit is only adhered to if strict overcommit accounting is enabled (mode 2 in `vm.overcommit_memory`). **CommitLimit** is calculated with the following formula:

$$\frac{([\text{total RAM pages}] - [\text{total huge TLB pages}]) * \text{overcommit\_ratio}}{[\text{total swap pages}] + 100}$$

For example, on a system with 1 GB of physical RAM and 7 GB of swap with a `vm.overcommit_ratio` of 30 it would yield a **CommitLimit** of 7.3 GB.

- **Committed\_AS** — The total amount of memory, in kibibytes, estimated to complete the workload. This value represents the worst case scenario value, and also includes swap memory.
- **VMallocTotal** — The total amount of memory, in kibibytes, of total allocated virtual address space.
- **VMallocUsed** — The total amount of memory, in kibibytes, of used virtual address space.
- **VMallocChunk** — The largest contiguous block of memory, in kibibytes, of available virtual address space.
- **HardwareCorrupted** — The amount of memory, in kibibytes, with physical memory corruption problems, identified by the hardware and set aside by the kernel so it does not get used.
- **AnonHugePages** — The total amount of memory, in kibibytes, used by huge pages that are not backed by files and are mapped into userspace page tables.
- **HugePages\_Total** — The total number of hugepages for the system. The number is derived by dividing **Hugepagesize** by the megabytes set aside for hugepages specified in `/proc/sys/vm/hugetlb_pool`. *This statistic only appears on the x86, Itanium, and AMD64 architectures.*
- **HugePages\_Free** — The total number of hugepages available for the system. *This statistic only appears on the x86, Itanium, and AMD64 architectures.*
- **HugePages\_Rsvd** — The number of unused huge pages reserved for hugetlbfs.
- **HugePages\_Surp** — The number of surplus huge pages.
- **Hugepagesize** — The size for each hugepages unit in kibibytes. By default, the value is 4096 KB on uniprocessor kernels for 32 bit architectures. For SMP, hugemem kernels, and AMD64, the default is 2048 KB. For Itanium architectures, the default is 262144 KB. *This statistic only appears on the x86, Itanium, and AMD64 architectures.*
- **DirectMap4k** — The amount of memory, in kibibytes, mapped into kernel address space with 4 kB page mappings.

- **DirectMap2M** — The amount of memory, in kibibytes, mapped into kernel address space with 2 MB page mappings.

### E.2.19. /proc/misc

This file lists miscellaneous drivers registered on the miscellaneous major device, which is device number 10:

```
63 device-mapper 175 agpgart 135 rtc 134 apm_bios
```

The first column is the minor number of each device, while the second column shows the driver in use.

### E.2.20. /proc/modules

This file displays a list of all modules loaded into the kernel. Its contents vary based on the configuration and use of your system, but it should be organized in a similar manner to this sample **/proc/modules** file output:



#### NOTE

This example has been reformatted into a readable format. Most of this information can also be viewed via the **/sbin/lsmmod** command.

```
nfs      170109  0 -          Live 0x129b0000
lockd    51593    1 nfs,      Live 0x128b0000
nls_utf8 1729     0 -          Live 0x12830000
vfat     12097    0 -          Live 0x12823000
fat      38881    1 vfat,     Live 0x1287b000
autofs4  20293    2 -          Live 0x1284f000
sunrpc   140453   3 nfs,lockd, Live 0x12954000
3c59x    33257    0 -          Live 0x12871000
uhci_hcd 28377    0 -          Live 0x12869000
md5      3777     1 -          Live 0x1282c000
ipv6     211845  16 -          Live 0x128de000
ext3     92585    2 -          Live 0x12886000
jbd      65625    1 ext3,     Live 0x12857000
dm_mod   46677    3 -          Live 0x12833000
```

The first column contains the name of the module.

The second column refers to the memory size of the module, in bytes.

The third column lists how many instances of the module are currently loaded. A value of zero represents an unloaded module.

The fourth column states if the module depends upon another module to be present in order to function, and lists those other modules.

The fifth column lists what load state the module is in: **Live**, **Loading**, or **Unloading** are the only possible values.

The sixth column lists the current kernel memory offset for the loaded module. This information can be useful for debugging purposes, or for profiling tools such as **oprofile**.

### E.2.21. /proc/mounts

This file provides a list of all mounts in use by the system:

```
rootfs / rootfs rw 0 0
/proc /proc proc rw,nodiratime 0 0 none
/dev ramfs rw 0 0
/dev/mapper/VolGroup00-LogVol00 / ext3 rw 0 0
none /dev ramfs rw 0 0
/proc /proc proc rw,nodiratime 0 0
/sys /sys sysfs rw 0 0
none /dev/pts devpts rw 0 0
usbdevfs /proc/bus/usb usbdevfs rw 0 0
/dev/hda1 /boot ext3 rw 0 0
none /dev/shm tmpfs rw 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
sunrpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw 0 0
```

The output found here is similar to the contents of `/etc/mtab`, except that `/proc/mounts` is more up-to-date.

The first column specifies the device that is mounted, the second column reveals the mount point, and the third column tells the file system type, and the fourth column tells you if it is mounted read-only (**ro**) or read-write (**rw**). The fifth and sixth columns are dummy values designed to match the format used in `/etc/mtab`.

### E.2.22. /proc/mtrr

This file refers to the current Memory Type Range Registers (MTRRs) in use with the system. If the system architecture supports MTRRs, then the `/proc/mtrr` file may look similar to the following:

```
reg00: base=0x00000000 ( 0MB), size= 256MB: write-back, count=1
reg01: base=0xe8000000 (3712MB), size= 32MB: write-combining, count=1
```

MTRRs are used with the Intel P6 family of processors (Pentium II and higher) and control processor access to memory ranges. When using a video card on a PCI or AGP bus, a properly configured `/proc/mtrr` file can increase performance more than 150%.

Most of the time, this value is properly configured by default. More information on manually configuring this file can be found locally at the following location:

```
/usr/share/doc/kernel-doc-<kernel_version>/Documentation/<arch>/mtrr.txt
```

### E.2.23. /proc/partitions

This file contains partition block allocation information. A sampling of this file from a basic system looks similar to the following:

```
major minor #blocks name
 3      0 19531250 hda
 3      1  104391 hda1
 3      2 19422585 hda2
253     0 22708224 dm-0
253     1  524288 dm-1
```

Most of the information here is of little importance to the user, except for the following columns:

- **major** — The major number of the device with this partition. The major number in the `/proc/partitions`, (3), corresponds with the block device `ide0`, in `/proc/devices`.
- **minor** — The minor number of the device with this partition. This serves to separate the partitions into different physical devices and relates to the number at the end of the name of the partition.
- **#blocks** — Lists the number of physical disk blocks contained in a particular partition.
- **name** — The name of the partition.

## E.2.24. `/proc/slabinfo`

This file gives full information about memory usage on the *slab* level. Linux kernels greater than version 2.2 use *slab pools* to manage memory above the page level. Commonly used objects have their own slab pools.

Instead of parsing the highly verbose `/proc/slabinfo` file manually, the `/usr/bin/slabtop` program displays kernel slab cache information in real time. This program allows for custom configurations, including column sorting and screen refreshing.

A sample screen shot of `/usr/bin/slabtop` usually looks like the following example:

```
Active / Total Objects (% used)      : 133629 / 147300 (90.7%)
Active / Total Slabs (% used)        : 11492 / 11493 (100.0%)
Active / Total Caches (% used)       : 77 / 121 (63.6%)
Active / Total Size (% used)         : 41739.83K / 44081.89K (94.7%)
Minimum / Average / Maximum Object  : 0.01K / 0.30K / 128.00K
OBJS  ACTIVE USE    OBJ  SIZE    SLABS OBJ/SLAB CACHE SIZE NAME
44814 43159 96%   0.62K 7469     6   29876K ext3_inode_cache
36900 34614 93%   0.05K  492    75   1968K  buffer_head
35213 33124 94%   0.16K 1531    23   6124K dentry_cache
7364  6463 87%   0.27K  526    14   2104K radix_tree_node
2585  1781 68%   0.08K  55     47   220K  vm_area_struct
2263  2116 93%   0.12K  73     31   292K  size-128
1904  1125 59%   0.03K  16    119    64K  size-32
1666   768 46%   0.03K  14    119    56K  anon_vma
1512  1482 98%   0.44K 168     9   672K  inode_cache
1464  1040 71%   0.06K  24    61    96K  size-64
1320   820 62%   0.19K  66    20   264K  filp
678   587 86%   0.02K  3    226   12K  dm_io
678   587 86%   0.02K  3    226   12K  dm_tio
576   574 99%   0.47K  72     8   288K  proc_inode_cache
528   514 97%   0.50K  66     8   264K  size-512
492   372 75%   0.09K  12    41    48K  bio
465   314 67%   0.25K  31    15   124K  size-256
452   331 73%   0.02K  2    226    8K  biovec-1
420   420 100%  0.19K  21    20    84K  skbuff_head_cache
305   256 83%   0.06K  5     61    20K  biovec-4
290    4  1%   0.01K  1    290    4K  revoke_table
```

264	264	100%	4.00K	264	1	1056K	size-4096
260	256	98%	0.19K	13	20	52K	biovec-16
260	256	98%	0.75K	52	5	208K	biovec-64

Some of the more commonly used statistics in `/proc/slabinfo` that are included into `/usr/bin/slabtop` include:

- **OBJS** — The total number of objects (memory blocks), including those in use (allocated), and some spares not in use.
- **ACTIVE** — The number of objects (memory blocks) that are in use (allocated).
- **USE** — Percentage of total objects that are active.  $((ACTIVE/OBJS)(100))$
- **OBJ SIZE** — The size of the objects.
- **SLABS** — The total number of slabs.
- **OBJ/SLAB** — The number of objects that fit into a slab.
- **CACHE SIZE** — The cache size of the slab.
- **NAME** — The name of the slab.

For more information on the `/usr/bin/slabtop` program, refer to the `slabtop` man page.

## E.2.25. `/proc/stat`

This file keeps track of a variety of different statistics about the system since it was last restarted. The contents of `/proc/stat`, which can be quite long, usually begins like the following example:

```
cpu 259246 7001 60190 34250993 137517 772 0
cpu0 259246 7001 60190 34250993 137517 772 0
intr 354133732 347209999 2272 0 4 4 0 0 3 1 1249247 0 0 80143 0 422626
5169433
ctxt 12547729
btime 1093631447
processes 130523
procs_running 1
procs_blocked 0
preempt 5651840
cpu 209841 1554 21720 118519346 72939 154 27168
cpu0 42536 798 4841 14790880 14778 124 3117
cpu1 24184 569 3875 14794524 30209 29 3130
cpu2 28616 11 2182 14818198 4020 1 3493
cpu3 35350 6 2942 14811519 3045 0 3659
cpu4 18209 135 2263 14820076 12465 0 3373
cpu5 20795 35 1866 14825701 4508 0 3615
cpu6 21607 0 2201 14827053 2325 0 3334
cpu7 18544 0 1550 14831395 1589 0 3447
intr 15239682 14857833 6 0 6 6 0 5 0 1 0 0 0 29 0 2 0 0 0 0 0 0 0 94982 0
286812
ctxt 4209609
btime 1078711415
```

```
processes 21905
procs_running 1
procs_blocked 0
```

Some of the more commonly used statistics include:

- **cpu** — Measures the number of *jiffies* (1/100 of a second for x86 systems) that the system has been in user mode, user mode with low priority (*nice*), system mode, idle task, I/O wait, IRQ (*hardirq*), and *softirq* respectively. The IRQ (*hardirq*) is the direct response to a hardware event. The IRQ takes minimal work for queuing the "heavy" work up for the *softirq* to execute. The *softirq* runs at a lower priority than the IRQ and therefore may be interrupted more frequently. The total for all CPUs is given at the top, while each individual CPU is listed below with its own statistics. The following example is a 4-way Intel Pentium Xeon configuration with multi-threading enabled, therefore showing four physical processors and four virtual processors totaling eight processors.
- **page** — The number of memory pages the system has written in and out to disk.
- **swap** — The number of swap pages the system has brought in and out.
- **intr** — The number of interrupts the system has experienced.
- **btime** — The boot time, measured in the number of seconds since January 1, 1970, otherwise known as the *epoch*.

### E.2.26. /proc/swaps

This file measures swap space and its utilization. For a system with only one swap partition, the output of **/proc/swaps** may look similar to the following:

```
Filename                                     Type      Size      Used      Priority
/dev/mapper/VolGroup00-LogVol01             partition 524280    0         -1
```

While some of this information can be found in other files in the **/proc/** directory, **/proc/swap** provides a snapshot of every swap file name, the type of swap space, the total size, and the amount of space in use (in kilobytes). The priority column is useful when multiple swap files are in use. The lower the priority, the more likely the swap file is to be used.

### E.2.27. /proc/sysrq-trigger

Using the **echo** command to write to this file, a remote root user can execute most System Request Key commands remotely as if at the local terminal. To **echo** values to this file, the **/proc/sys/kernel/sysrq** must be set to a value other than **0**. For more information about the System Request Key, see [Section E.3.9.3, "/proc/sys/kernel/"](#).

Although it is possible to write to this file, it cannot be read, even by the root user.

### E.2.28. /proc/uptime

This file contains information detailing how long the system has been on since its last restart. The output of **/proc/uptime** is quite minimal:

```
350735.47 234388.90
```

The first value represents the total number of seconds the system has been up. The second value is the sum of how much time each core has spent idle, in seconds. Consequently, the second value may be greater than the overall system uptime on systems with multiple cores.

## E.2.29. `/proc/version`

This file specifies the version of the Linux kernel, the version of **gcc** used to compile the kernel, and the time of kernel compilation. It also contains the kernel compiler's user name (in parentheses).

```
Linux version 2.6.8-1.523 (user@foo.redhat.com) (gcc version 3.4.1
20040714 \ (Red Hat Enterprise Linux 3.4.1-7)) #1 Mon Aug 16 13:27:03 EDT
2004
```

This information is used for a variety of purposes, including the version data presented when a user logs in.

## E.3. DIRECTORIES WITHIN `/PROC/`

Common groups of information concerning the kernel are grouped into directories and subdirectories within the `/proc/` directory.

### E.3.1. Process Directories

Every `/proc/` directory contains a number of directories with numerical names. A listing of them may be similar to the following:

```
dr-xr-xr-x    3 root    root          0 Feb 13 01:28 1
dr-xr-xr-x    3 root    root          0 Feb 13 01:28 1010
dr-xr-xr-x    3 xfs     xfs           0 Feb 13 01:28 1087
dr-xr-xr-x    3 daemon  daemon       0 Feb 13 01:28 1123
dr-xr-xr-x    3 root    root          0 Feb 13 01:28 11307
dr-xr-xr-x    3 apache  apache       0 Feb 13 01:28 13660
dr-xr-xr-x    3 rpc     rpc           0 Feb 13 01:28 637
dr-xr-xr-x    3 rpcuser rpcuser      0 Feb 13 01:28 666
```

These directories are called *process directories*, as they are named after a program's process ID and contain information specific to that process. The owner and group of each process directory is set to the user running the process. When the process is terminated, its `/proc/` process directory vanishes.

Each process directory contains the following files:

- **cmdline** — Contains the command issued when starting the process.
- **cwd** — A symbolic link to the current working directory for the process.
- **environ** — A list of the environment variables for the process. The environment variable is given in all upper-case characters, and the value is in lower-case characters.
- **exe** — A symbolic link to the executable of this process.
- **fd** — A directory containing all of the file descriptors for a particular process. These are given in numbered links:

```
total 0
```

```

lrwx----- 1 root    root      64 May  8 11:31 0 ->
/dev/null
lrwx----- 1 root    root      64 May  8 11:31 1 ->
/dev/null
lrwx----- 1 root    root      64 May  8 11:31 2 ->
/dev/null
lrwx----- 1 root    root      64 May  8 11:31 3 ->
/dev/ptmx
lrwx----- 1 root    root      64 May  8 11:31 4 ->
socket:[7774817]
lrwx----- 1 root    root      64 May  8 11:31 5 ->
/dev/ptmx
lrwx----- 1 root    root      64 May  8 11:31 6 ->
socket:[7774829]
lrwx----- 1 root    root      64 May  8 11:31 7 ->
/dev/ptmx

```

- **maps** — A list of memory maps to the various executables and library files associated with this process. This file can be rather long, depending upon the complexity of the process, but sample output from the **sshd** process begins like the following:

```

08048000-08086000 r-xp 00000000 03:03 391479    /usr/sbin/sshd
08086000-08088000 rw-p 0003e000 03:03 391479    /usr/sbin/sshd
08088000-08095000 rwxp 00000000 00:00 0
40000000-40013000 r-xp 00000000 03:03 293205    /lib/ld-2.2.5.so
40013000-40014000 rw-p 00013000 03:03 293205    /lib/ld-2.2.5.so
40031000-40038000 r-xp 00000000 03:03 293282    /lib/libpam.so.0.75
40038000-40039000 rw-p 00006000 03:03 293282    /lib/libpam.so.0.75
40039000-4003a000 rw-p 00000000 00:00 0
4003a000-4003c000 r-xp 00000000 03:03 293218    /lib/libdl-2.2.5.so
4003c000-4003d000 rw-p 00001000 03:03 293218    /lib/libdl-2.2.5.so

```

- **mem** — The memory held by the process. This file cannot be read by the user.
- **root** — A link to the root directory of the process.
- **stat** — The status of the process.
- **statm** — The status of the memory in use by the process. Below is a sample **/proc/statm** file:

```

263 210 210 5 0 205 0

```

The seven columns relate to different memory statistics for the process. From left to right, they report the following aspects of the memory used:

1. Total program size, in kilobytes.
2. Size of memory portions, in kilobytes.
3. Number of pages that are shared.
4. Number of pages that are code.
5. Number of pages of data/stack.



6. Number of library pages.
  7. Number of dirty pages.
- **status** — The status of the process in a more readable form than **stat** or **statm**. Sample output for **sshd** looks similar to the following:

```
Name: sshd
State: S (sleeping)
Tgid: 797
Pid: 797
PPid: 1
TracerPid: 0
Uid: 0 0 0 0
Gid: 0 0 0 0
FDSize: 32
Groups:
VmSize:      3072 kB
VmLck:        0 kB
VmRSS:       840 kB
VmData:      104 kB
VmStk:        12 kB
VmExe:       300 kB
VmLib:       2528 kB
SigPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 80000000000001000
SigCgt: 00000000000014005
CapInh: 0000000000000000
CapPrm: 00000000ffffffeff
CapEff: 00000000ffffffeff
```

The information in this output includes the process name and ID, the state (such as **S (sleeping)** or **R (running)**), user/group ID running the process, and detailed data regarding memory usage.

### E.3.1.1. `/proc/self/`

The `/proc/self/` directory is a link to the currently running process. This allows a process to look at itself without having to know its process ID.

Within a shell environment, a listing of the `/proc/self/` directory produces the same contents as listing the process directory for that process.

### E.3.2. `/proc/bus/`

This directory contains information specific to the various buses available on the system. For example, on a standard system containing PCI and USB buses, current data on each of these buses is available within a subdirectory within `/proc/bus/` by the same name, such as `/proc/bus/pci/`.

The subdirectories and files available within `/proc/bus/` vary depending on the devices connected to the system. However, each bus type has at least one directory. Within these bus directories are normally at least one subdirectory with a numerical name, such as `001`, which contain binary files.

For example, the `/proc/bus/usb/` subdirectory contains files that track the various devices on any USB buses, as well as the drivers required for them. The following is a sample listing of a `/proc/bus/usb/` directory:

```
total 0 dr-xr-xr-x    1 root    root                0 May  3 16:25 001
-r--r--r--    1 root    root                0 May  3 16:25 devices
-r--r--r--    1 root    root                0 May  3 16:25 drivers
```

The `/proc/bus/usb/001/` directory contains all devices on the first USB bus and the `devices` file identifies the USB root hub on the motherboard.

The following is an example of a `/proc/bus/usb/devices` file:

```
T: Bus=01 Lev=00 Prnt=00 Port=00 Cnt=00 Dev#= 1 Spd=12 MxCh= 2
B: Alloc= 0/900 us ( 0%), #Int= 0, #Iso= 0
D: Ver= 1.00 Cls=09(hub ) Sub=00 Prot=00 MxPS= 8 #Cfgs= 1
P: Vendor=0000 ProdID=0000 Rev= 0.00
S: Product=USB UHCI Root Hub
S: SerialNumber=d400
C:* #Ifs= 1 Cfg#= 1 Atr=40 MxPwr= 0mA
I: If#= 0 Alt= 0 #EPs= 1 Cls=09(hub ) Sub=00 Prot=00 Driver=hub
E: Ad=81(I) Atr=03(Int.) MxPS= 8 Iv1=255ms
```

### E.3.3. `/proc/bus/pci`

Later versions of the 2.6 Linux kernel have obsoleted the `/proc/pci` directory in favor of the `/proc/bus/pci` directory. Although you can get a list of all PCI devices present on the system using the command `cat /proc/bus/pci/devices`, the output is difficult to read and interpret.

For a human-readable list of PCI devices, run the following command:

```
~]# /sbin/lspci -vb
00:00.0 Host bridge: Intel Corporation 82X38/X48 Express DRAM Controller
Subsystem: Hewlett-Packard Company Device 1308
Flags: bus master, fast devsel, latency 0
Capabilities: [e0] Vendor Specific Information <?>
Kernel driver in use: x38_edac
Kernel modules: x38_edac

00:01.0 PCI bridge: Intel Corporation 82X38/X48 Express Host-Primary PCI
Express Bridge (prog-if 00 [Normal decode])
Flags: bus master, fast devsel, latency 0
Bus: primary=00, secondary=01, subordinate=01, sec-latency=0
I/O behind bridge: 00001000-00001fff
Memory behind bridge: f0000000-f2ffffff
Capabilities: [88] Subsystem: Hewlett-Packard Company Device 1308
Capabilities: [80] Power Management version 3
Capabilities: [90] MSI: Enable+ Count=1/1 Maskable- 64bit-
Capabilities: [a0] Express Root Port (Slot+), MSI 00
Capabilities: [100] Virtual Channel <?>
Capabilities: [140] Root Complex Link <?>
Kernel driver in use: pcieport
Kernel modules: shpchp
```

```
00:1a.0 USB Controller: Intel Corporation 82801I (ICH9 Family) USB UHCI
Controller #4 (rev 02) (prog-if 00 [UHCI])
    Subsystem: Hewlett-Packard Company Device 1308
    Flags: bus master, medium devsel, latency 0, IRQ 5
    I/O ports at 2100
    Capabilities: [50] PCI Advanced Features
    Kernel driver in use: uhci_hcd
[output truncated]
```

The output is a sorted list of all IRQ numbers and addresses as seen by the cards on the PCI bus instead of as seen by the kernel. Beyond providing the name and version of the device, this list also gives detailed IRQ information so an administrator can quickly look for conflicts.

### E.3.4. `/proc/driver/`

This directory contains information for specific drivers in use by the kernel.

A common file found here is `rtc` which provides output from the driver for the system's *Real Time Clock (RTC)*, the device that keeps the time while the system is switched off. Sample output from `/proc/driver/rtc` looks like the following:

```
rtc_time      : 16:21:00
rtc_date      : 2004-08-31
rtc_epoch     : 1900
alarm        : 21:16:27
DST_enable    : no
BCD          : yes
24hr         : yes
square_wave   : no
alarm_IRQ    : no
update_IRQ   : no
periodic_IRQ  : no
periodic_freq : 1024
batt_status   : okay
```

For more information about the RTC, see the following installed documentation:

`/usr/share/doc/kernel-doc-<kernel_version>/Documentation/rtc.txt.`

### E.3.5. `/proc/fs`

This directory shows which file systems are exported. If running an NFS server, typing `cat /proc/fs/nfsd/exports` displays the file systems being shared and the permissions granted for those file systems. For more on file system sharing with NFS, see the *Network File System (NFS)* chapter of the *Storage Administration Guide*.

### E.3.6. `/proc/irq/`

This directory is used to set IRQ to CPU affinity, which allows the system to connect a particular IRQ to only one CPU. Alternatively, it can exclude a CPU from handling any IRQs.

Each IRQ has its own directory, allowing for the individual configuration of each IRQ. The `/proc/irq/prof_cpu_mask` file is a bitmask that contains the default values for the `smp_affinity` file in the IRQ directory. The values in `smp_affinity` specify which CPUs handle that particular IRQ.

For more information about the `/proc/irq/` directory, see the following installed documentation:

```
/usr/share/doc/kernel-  
doc-kernel_version/Documentation/filesystems/proc.txt
```

### E.3.7. `/proc/net/`

This directory provides a comprehensive look at various networking parameters and statistics. Each directory and virtual file within this directory describes aspects of the system's network configuration. Below is a partial list of the `/proc/net/` directory:

- **arp** — Lists the kernel's ARP table. This file is particularly useful for connecting a hardware address to an IP address on a system.
- **atm/** directory — The files within this directory contain *Asynchronous Transfer Mode (ATM)* settings and statistics. This directory is primarily used with ATM networking and ADSL cards.
- **dev** — Lists the various network devices configured on the system, complete with transmit and receive statistics. This file displays the number of bytes each interface has sent and received, the number of packets inbound and outbound, the number of errors seen, the number of packets dropped, and more.
- **dev\_mcast** — Lists Layer2 multicast groups on which each device is listening.
- **igmp** — Lists the IP multicast addresses which this system joined.
- **ip\_contrack** — Lists tracked network connections for machines that are forwarding IP connections.
- **ip\_tables\_names** — Lists the types of **iptables** in use. This file is only present if **iptables** is active on the system and contains one or more of the following values: **filter**, **mangle**, or **nat**.
- **ip\_mr\_cache** — Lists the multicast routing cache.
- **ip\_mr\_vif** — Lists multicast virtual interfaces.
- **netstat** — Contains a broad yet detailed collection of networking statistics, including TCP timeouts, SYN cookies sent and received, and much more.
- **psched** — Lists global packet scheduler parameters.
- **raw** — Lists raw device statistics.
- **route** — Lists the kernel's routing table.
- **rt\_cache** — Contains the current routing cache.
- **snmp** — List of Simple Network Management Protocol (SNMP) data for various networking protocols in use.
- **sockstat** — Provides socket statistics.
- **tcp** — Contains detailed TCP socket information.

- **tr\_rif** — Lists the token ring RIF routing table.
- **udp** — Contains detailed UDP socket information.
- **unix** — Lists UNIX domain sockets currently in use.
- **wireless** — Lists wireless interface data.

### E.3.8. /proc/scsi/

The primary file in this directory is **/proc/scsi/scsi**, which contains a list of every recognized SCSI device. From this listing, the type of device, as well as the model name, vendor, SCSI channel and ID data is available.

For example, if a system contains a SCSI CD-ROM, a tape drive, a hard drive, and a RAID controller, this file looks similar to the following:

```
Attached devices:
Host: scsi1
Channel: 00
Id: 05
Lun: 00
Vendor: NEC
Model: CD-ROM DRIVE:466
Rev: 1.06
Type:    CD-ROM
ANSI SCSI revision: 02
Host: scsi1
Channel: 00
Id: 06
Lun: 00
Vendor: ARCHIVE
Model: Python 04106-XXX
Rev: 7350
Type:    Sequential-Access
ANSI SCSI revision: 02
Host: scsi2
Channel: 00
Id: 06
Lun: 00
Vendor: DELL
Model: 1x6 U2W SCSI BP
Rev: 5.35
Type:    Processor
ANSI SCSI revision: 02
Host: scsi2
Channel: 02
Id: 00
Lun: 00
Vendor: MegaRAID
Model: LD0 RAID5 34556R
Rev: 1.01
Type:    Direct-Access
ANSI SCSI revision: 02
```

Each SCSI driver used by the system has its own directory within `/proc/scsi/`, which contains files specific to each SCSI controller using that driver. From the previous example, `aic7xxx/` and `megaraid/` directories are present, since two drivers are in use. The files in each of the directories typically contain an I/O address range, IRQ information, and statistics for the SCSI controller using that driver. Each controller can report a different type and amount of information. The Adaptec AIC-7880 Ultra SCSI host adapter's file in this example system produces the following output:

```

Adaptec AIC7xxx driver version: 5.1.20/3.2.4
Compile Options:
TCQ Enabled By Default : Disabled
AIC7XXX_PROC_STATS      : Enabled
AIC7XXX_RESET_DELAY    : 5
Adapter Configuration:
SCSI Adapter: Adaptec AIC-7880 Ultra SCSI host adapter
Ultra Narrow Controller      PCI MMAPed
I/O Base: 0xfcffe000
Adapter SEEPROM Config: SEEPROM found and used.
Adaptec SCSI BIOS: Enabled
IRQ: 30
SCBs: Active 0, Max Active 1, Allocated 15, HW 16, Page 255
Interrupts: 33726
BIOS Control Word: 0x18a6
Adapter Control Word: 0x1c5f
Extended Translation: Enabled
Disconnect Enable Flags: 0x00ff
Ultra Enable Flags: 0x0020
Tag Queue Enable Flags: 0x0000
Ordered Queue Tag Flags: 0x0000
Default Tag Queue Depth: 8
Tagged Queue By Device array for aic7xxx
host instance 1:
{255,255,255,255,255,255,255,255,255,255,255,255,255,255}
Actual queue depth per device for aic7xxx host instance 1:
{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}
Statistics:

(scsi1:0:5:0) Device using Narrow/Sync transfers at 20.0 MByte/sec, offset
15
Transinfo settings: current(12/15/0/0), goal(12/15/0/0), user(12/15/0/0)
Total transfers 0 (0 reads and 0 writes)
  < 2K    2K+    4K+    8K+    16K+    32K+    64K+    128K+
Reads:      0      0      0      0      0      0      0      0
Writes:     0      0      0      0      0      0      0      0

(scsi1:0:6:0) Device using Narrow/Sync transfers at 10.0 MByte/sec, offset
15
Transinfo settings: current(25/15/0/0), goal(12/15/0/0), user(12/15/0/0)
Total transfers 132 (0 reads and 132 writes)
  < 2K    2K+    4K+    8K+    16K+    32K+    64K+    128K+
Reads:      0      0      0      0      0      0      0      0
Writes:     0      0      0      0      1     131     0      0

```

This output reveals the transfer speed to the SCSI devices connected to the controller based on channel ID, as well as detailed statistics concerning the amount and sizes of files read or written by that device. For example, this controller is communicating with the CD-ROM at 20 megabytes per second, while the tape drive is only communicating at 10 megabytes per second.

### E.3.9. /proc/sys/

The **/proc/sys/** directory is different from others in **/proc/** because it not only provides information about the system but also allows the system administrator to immediately enable and disable kernel features.



#### WARNING

Use caution when changing settings on a production system using the various files in the **/proc/sys/** directory. Changing the wrong setting may render the kernel unstable, requiring a system reboot.

For this reason, be sure the options are valid for that file before attempting to change any value in **/proc/sys/**.

A good way to determine if a particular file can be configured, or if it is only designed to provide information, is to list it with the **-l** option at the shell prompt. If the file is writable, it may be used to configure the kernel. For example, a partial listing of **/proc/sys/fs** looks like the following:

```
-r--r--r--    1 root    root          0 May 10 16:14 dentry-state
-rw-r--r--    1 root    root          0 May 10 16:14 dir-notify-enable
-rw-r--r--    1 root    root          0 May 10 16:14 file-max
-r--r--r--    1 root    root          0 May 10 16:14 file-nr
```

In this listing, the files **dir-notify-enable** and **file-max** can be written to and, therefore, can be used to configure the kernel. The other files only provide feedback on current settings.

Changing a value within a **/proc/sys/** file is done by echoing the new value into the file. For example, to enable the System Request Key on a running kernel, type the command:

```
echo 1 > /proc/sys/kernel/sysrq
```

This changes the value for **sysrq** from **0** (off) to **1** (on).

A few **/proc/sys/** configuration files contain more than one value. To correctly send new values to them, place a space character between each value passed with the **echo** command, such as is done in this example:

```
echo 4 2 45 > /proc/sys/kernel/acct
```



#### NOTE

Any configuration changes made using the **echo** command disappear when the system is restarted. To make configuration changes take effect after the system is rebooted, see [Section E.4, “Using the sysctl Command”](#).

The **/proc/sys/** directory contains several subdirectories controlling different aspects of a running kernel.

### E.3.9.1. /proc/sys/dev/

This directory provides parameters for particular devices on the system. Most systems have at least two directories, **cdrom/** and **raid/**. Customized kernels can have other directories, such as **parport/**, which provides the ability to share one parallel port between multiple device drivers.

The **cdrom/** directory contains a file called **info**, which reveals a number of important CD-ROM parameters:

```
CD-ROM information, Id: cdrom.c 3.20 2003/12/17
drive name:          hdc
drive speed:         48
drive # of slots:    1
Can close tray:      1
Can open tray:       1
Can lock tray:       1
Can change speed:    1
Can select disk:     0
Can read multisession: 1
Can read MCN:        1
Reports media changed: 1
Can play audio:      1
Can write CD-R:      0
Can write CD-RW:     0
Can read DVD:        0
Can write DVD-R:     0
Can write DVD-RAM:   0
Can read MRW:        0
Can write MRW:       0
Can write RAM:       0
```

This file can be quickly scanned to discover the qualities of an unknown CD-ROM. If multiple CD-ROMs are available on a system, each device is given its own column of information.

Various files in **/proc/sys/dev/cdrom**, such as **autoclose** and **checkmedia**, can be used to control the system's CD-ROM. Use the **echo** command to enable or disable these features.

If RAID support is compiled into the kernel, a **/proc/sys/dev/raid/** directory becomes available with at least two files in it: **speed\_limit\_min** and **speed\_limit\_max**. These settings determine the acceleration of RAID devices for I/O intensive tasks, such as resyncing the disks.

### E.3.9.2. /proc/sys/fs/

This directory contains an array of options and information concerning various aspects of the file system, including quota, file handle, inode, and dentry information.

The **binfmt\_misc/** directory is used to provide kernel support for miscellaneous binary formats.

The important files in **/proc/sys/fs/** include:

- **dentry-state** — Provides the status of the directory cache. The file looks similar to the following:

```
57411 52939 45 0 0 0
```



The first number reveals the total number of directory cache entries, while the second number displays the number of unused entries. The third number tells the number of seconds between when a directory has been freed and when it can be reclaimed, and the fourth measures the pages currently requested by the system. The last two numbers are not used and display only zeros.

- **file-max** — Lists the maximum number of file handles that the kernel allocates. Raising the value in this file can resolve errors caused by a lack of available file handles.
- **file-nr** — Lists the number of allocated file handles, used file handles, and the maximum number of file handles.
- **overflowgid** and **overflowuid** — Defines the fixed group ID and user ID, respectively, for use with file systems that only support 16-bit group and user IDs.

### E.3.9.3. /proc/sys/kernel/

This directory contains a variety of different configuration files that directly affect the operation of the kernel. Some of the most important files include:

- **acct** — Controls the suspension of process accounting based on the percentage of free space available on the file system containing the log. By default, the file looks like the following:

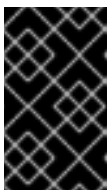
```
4 2 30
```

The first value dictates the percentage of free space required for logging to resume, while the second value sets the threshold percentage of free space when logging is suspended. The third value sets the interval, in seconds, that the kernel polls the file system to see if logging should be suspended or resumed.

- **ctrl-alt-del** — Controls whether **Ctrl+Alt+Delete** gracefully restarts the computer using **init (0)** or forces an immediate reboot without syncing the dirty buffers to disk (**1**).
- **domainname** — Configures the system domain name, such as **example.com**.
- **exec-shield** — Configures the Exec Shield feature of the kernel. Exec Shield provides protection against certain types of buffer overflow attacks.

There are two possible values for this virtual file:

- **0** — Disables Exec Shield.
- **1** — Enables Exec Shield. This is the default value.



#### IMPORTANT

If a system is running security-sensitive applications that were started while Exec Shield was disabled, these applications must be restarted when Exec Shield is enabled in order for Exec Shield to take effect.

- **hostname** — Configures the system host name, such as **www.example.com**.

- **hotplug** — Configures the utility to be used when a configuration change is detected by the system. This is primarily used with USB and Cardbus PCI. The default value of `/sbin/hotplug` should not be changed unless testing a new program to fulfill this role.
- **modprobe** — Sets the location of the program used to load kernel modules. The default value is `/sbin/modprobe` which means **kmod** calls it to load the module when a kernel thread calls **kmod**.
- **msgmax** — Sets the maximum size of any message sent from one process to another and is set to **8192** bytes by default. Be careful when raising this value, as queued messages between processes are stored in non-swappable kernel memory. Any increase in **msgmax** would increase RAM requirements for the system.
- **msgmnb** — Sets the maximum number of bytes in a single message queue. The default is **16384**.
- **msgmni** — Sets the maximum number of message queue identifiers. The default is **4008**.
- **osrelease** — Lists the Linux kernel release number. This file can only be altered by changing the kernel source and recompiling.
- **ostype** — Displays the type of operating system. By default, this file is set to **Linux**, and this value can only be changed by changing the kernel source and recompiling.
- **overflowgid** and **overflowuid** — Defines the fixed group ID and user ID, respectively, for use with system calls on architectures that only support 16-bit group and user IDs.
- **panic** — Defines the number of seconds the kernel postpones rebooting when the system experiences a kernel panic. By default, the value is set to **0**, which disables automatic rebooting after a panic.
- **printk** — This file controls a variety of settings related to printing or logging error messages. Each error message reported by the kernel has a *loglevel* associated with it that defines the importance of the message. The loglevel values break down in this order:
  - **0** — Kernel emergency. The system is unusable.
  - **1** — Kernel alert. Action must be taken immediately.
  - **2** — Condition of the kernel is considered critical.
  - **3** — General kernel error condition.
  - **4** — General kernel warning condition.
  - **5** — Kernel notice of a normal but significant condition.
  - **6** — Kernel informational message.
  - **7** — Kernel debug-level messages.

Four values are found in the **printk** file:

```
6      4      1      7
```

Each of these values defines a different rule for dealing with error messages. The first value, called the *console loglevel*, defines the lowest priority of messages printed to the console. (Note that, the lower the priority, the higher the loglevel number.) The second value sets the default loglevel for messages without an explicit loglevel attached to them. The third value sets the lowest possible loglevel configuration for the console loglevel. The last value sets the default value for the console loglevel.

- **random/** directory — Lists a number of values related to generating random numbers for the kernel.
- **sem** — Configures *semaphore* settings within the kernel. A semaphore is a System V IPC object that is used to control utilization of a particular process.
- **shmall** — Sets the total amount of shared memory that can be used at one time on the system, in bytes. By default, this value is **2097152**.
- **shmmax** — Sets the largest shared memory segment size allowed by the kernel. By default, this value is **33554432**. However, the kernel supports much larger values than this.
- **shmmni** — Sets the maximum number of shared memory segments for the whole system. By default, this value is **4096**.
- **sysrq** — Activates the System Request Key, if this value is set to anything other than zero (**0**), the default.

The System Request Key allows immediate input to the kernel through simple key combinations. For example, the System Request Key can be used to immediately shut down or restart a system, sync all mounted file systems, or dump important information to the console. To initiate a System Request Key, type **Alt+SysRq+system request code**. Replace *system request code* with one of the following system request codes:

- **r** — Disables raw mode for the keyboard and sets it to XLATE (a limited keyboard mode which does not recognize modifiers such as **Alt**, **Ctrl**, or **Shift** for all keys).
- **k** — Kills all processes active in a virtual console. Also called *Secure Access Key (SAK)*, it is often used to verify that the login prompt is spawned from **init** and not a trojan copy designed to capture user names and passwords.
- **b** — Reboots the kernel without first unmounting file systems or syncing disks attached to the system.
- **c** — Crashes the system without first unmounting file systems or syncing disks attached to the system.
- **o** — Shuts off the system.
- **s** — Attempts to sync disks attached to the system.
- **u** — Attempts to unmount and remount all file systems as read-only.
- **p** — Outputs all flags and registers to the console.
- **t** — Outputs a list of processes to the console.
- **m** — Outputs memory statistics to the console.

- **0** through **9** — Sets the log level for the console.
- **e** — Kills all processes except **init** using SIGTERM.
- **i** — Kills all processes except **init** using SIGKILL.
- **l** — Kills all processes using SIGKILL (including **init**). *The system is unusable after issuing this System Request Key code.*
- **h** — Displays help text.

This feature is most beneficial when using a development kernel or when experiencing system freezes.



### WARNING

The System Request Key feature is considered a security risk because an unattended console provides an attacker with access to the system. For this reason, it is turned off by default.

See `/usr/share/doc/kernel-doc-kernel_version/Documentation/sysrq.txt` for more information about the System Request Key.

- **tainted** — Indicates whether a non-GPL module is loaded.
  - **0** — No non-GPL modules are loaded.
  - **1** — At least one module without a GPL license (including modules with no license) is loaded.
  - **2** — At least one module was force-loaded with the command **insmod -f**.
- **threads-max** — Sets the maximum number of threads to be used by the kernel, with a default value of **2048**.
- **version** — Displays the date and time the kernel was last compiled. The first field in this file, such as **#3**, relates to the number of times a kernel was built from the source base.

#### E.3.9.4. `/proc/sys/net/`

This directory contains subdirectories concerning various networking topics. Various configurations at the time of kernel compilation make different directories available here, such as **ethernet/**, **ipv4/**, **ipx/**, and **ipv6/**. By altering the files within these directories, system administrators are able to adjust the network configuration on a running system.

Given the wide variety of possible networking options available with Linux, only the most common `/proc/sys/net/` directories are discussed.

The `/proc/sys/net/core/` directory contains a variety of settings that control the interaction between the kernel and networking layers. The most important of these files are:

- **message\_burst** — Sets the maximum number of new warning messages to be written to the kernel log in the time interval defined by **message\_cost**. The default value of this file is **10**.

In combination with **message\_cost**, this setting is used to enforce a rate limit on warning messages written to the kernel log from the networking code and mitigate *Denial of Service* (DoS) attacks. The idea of a DoS attack is to bombard the targeted system with requests that generate errors and either fill up disk partitions with log files or require all of the system's resources to handle the error logging.

The settings in **message\_burst** and **message\_cost** are designed to be modified based on the system's acceptable risk versus the need for comprehensive logging. For example, by setting **message\_burst** to 10 and **message\_cost** to 5, you allow the system to write the maximum number of 10 messages every 5 seconds.

- **message\_cost** — Sets a cost on every warning message by defining a time interval for **message\_burst**. The higher the value is, the more likely the warning message is ignored. The default value of this file is **5**.
- **netdev\_max\_backlog** — Sets the maximum number of packets allowed to queue when a particular interface receives packets faster than the kernel can process them. The default value for this file is **1000**.
- **optmem\_max** — Configures the maximum ancillary buffer size allowed per socket.
- **rmem\_default** — Sets the receive socket buffer default size in bytes.
- **rmem\_max** — Sets the receive socket buffer maximum size in bytes.
- **wmem\_default** — Sets the send socket buffer default size in bytes.
- **wmem\_max** — Sets the send socket buffer maximum size in bytes.

The `/proc/sys/net/ipv4/` directory contains additional networking settings. Many of these settings, used in conjunction with one another, are useful in preventing attacks on the system or when using the system to act as a router.



### WARNING

An erroneous change to these files may affect remote connectivity to the system.

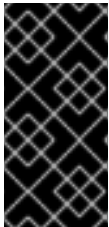
The following is a list of some of the more important files within the `/proc/sys/net/ipv4/` directory:

- **icmp\_echo\_ignore\_all** and **icmp\_echo\_ignore\_broadcasts** — Allows the kernel to ignore ICMP ECHO packets from every host or only those originating from broadcast and multicast addresses, respectively. A value of **0** allows the kernel to respond, while a value of **1** ignores the packets.
- **ip\_default\_ttl** — Sets the default *Time To Live (TTL)*, which limits the number of hops a packet may make before reaching its destination. Increasing this value can diminish system performance.

- **ip\_forward** — Permits interfaces on the system to forward packets. By default, this file is set to **0**. Setting this file to **1** enables network packet forwarding.
- **ip\_local\_port\_range** — Specifies the range of ports to be used by TCP or UDP when a local port is needed. The first number is the lowest port to be used and the second number specifies the highest port. Any systems that expect to require more ports than the default 1024 to 4999 should use a range from 32768 to 61000.
- **tcp\_syn\_retries** — Provides a limit on the number of times the system re-transmits a SYN packet when attempting to make a connection.
- **tcp\_retries1** — Sets the number of permitted re-transmissions attempting to answer an incoming connection. Default of **3**.
- **tcp\_retries2** — Sets the number of permitted re-transmissions of TCP packets. Default of **15**.

The `/usr/share/doc/kernel-doc-kernel_version/Documentation/networking/ip-sysctl.txt` file contains a list of files and options available in the `/proc/sys/net/ipv4/` and `/proc/sys/net/ipv6/` directories. Use the `sysctl -a` command to list the parameters in the `sysctl` key format.

A number of other directories exist within the `/proc/sys/net/ipv4/` directory and each covers a different aspect of the network stack. The `/proc/sys/net/ipv4/conf/` directory allows each system interface to be configured in different ways, including the use of default settings for unconfigured devices (in the `/proc/sys/net/ipv4/conf/default/` subdirectory) and settings that override all special configurations (in the `/proc/sys/net/ipv4/conf/all/` subdirectory).



### IMPORTANT

Red Hat Enterprise Linux 6 defaults to **strict reverse path forwarding**. Before changing the setting in the `rp_filter` file, see the entry on Reverse Path Forwarding in the [Red Hat Enterprise Linux 6 Security Guide](#) and [The Red Hat Knowledgebase article about rp\\_filter](#).

The `/proc/sys/net/ipv4/neigh/` directory contains settings for communicating with a host directly connected to the system (called a network neighbor) and also contains different settings for systems more than one hop away.

Routing over IPV4 also has its own directory, `/proc/sys/net/ipv4/route/`. Unlike `conf/` and `neigh/`, the `/proc/sys/net/ipv4/route/` directory contains specifications that apply to routing with any interfaces on the system. Many of these settings, such as `max_size`, `max_delay`, and `min_delay`, relate to controlling the size of the routing cache. To clear the routing cache, write any value to the `flush` file.

Additional information about these directories and the possible values for their configuration files can be found in:

```
/usr/share/doc/kernel-
doc-kernel_version/Documentation/filesystems/proc.txt
```

#### E.3.9.5. `/proc/sys/vm/`

This directory facilitates the configuration of the Linux kernel's virtual memory (VM) subsystem. The kernel makes extensive and intelligent use of virtual memory, which is commonly referred to as swap space.

The following files are commonly found in the `/proc/sys/vm/` directory:

- **block\_dump** — Configures block I/O debugging when enabled. All read/write and block dirtying operations done to files are logged accordingly. This can be useful if diagnosing disk spin up and spin downs for laptop battery conservation. All output when **block\_dump** is enabled can be retrieved via **dmesg**. The default value is **0**.



#### NOTE

If **block\_dump** is enabled at the same time as kernel debugging, it is prudent to stop the **klogd** daemon, as it generates erroneous disk activity caused by **block\_dump**.

- **dirty\_background\_ratio** — Starts background writeback of dirty data at this percentage of total memory, via a pdflush daemon. The default value is **10**.
- **dirty\_expire\_centisecs** — Defines when dirty in-memory data is old enough to be eligible for writeout. Data which has been dirty in-memory for longer than this interval is written out next time a pdflush daemon wakes up. The default value is **3000**, expressed in hundredths of a second.
- **dirty\_ratio** — Starts active writeback of dirty data at this percentage of total memory for the generator of dirty data, via pdflush. The default value is **20**.
- **dirty\_writeback\_centisecs** — Defines the interval between pdflush daemon wakeups, which periodically writes dirty in-memory data out to disk. The default value is **500**, expressed in hundredths of a second.
- **laptop\_mode** — Minimizes the number of times that a hard disk needs to spin up by keeping the disk spun down for as long as possible, therefore conserving battery power on laptops. This increases efficiency by combining all future I/O processes together, reducing the frequency of spin ups. The default value is **0**, but is automatically enabled in case a battery on a laptop is used.

This value is controlled automatically by the `acpid` daemon once a user is notified battery power is enabled. No user modifications or interactions are necessary if the laptop supports the ACPI (Advanced Configuration and Power Interface) specification.

For more information, see the following installed documentation:

`/usr/share/doc/kernel-doc-kernel_version/Documentation/laptop-mode.txt`

- **max\_map\_count** — Configures the maximum number of memory map areas a process may have. In most cases, the default value of **65536** is appropriate.
- **min\_free\_kbytes** — Forces the Linux VM (virtual memory manager) to keep a minimum number of kilobytes free. The VM uses this number to compute a **pages\_min** value for each **lowmem** zone in the system. The default value is in respect to the total memory on the machine.
- **nr\_hugepages** — Indicates the current number of configured **hugetlb** pages in the kernel.

For more information, see the following installed documentation:

**/usr/share/doc/kernel-doc-kernel\_version/Documentation/vm/hugetlbpage.txt**

- **nr\_pdflush\_threads** — Indicates the number of pdflush daemons that are currently running. This file is read-only, and should not be changed by the user. Under heavy I/O loads, the default value of two is increased by the kernel.
- **overcommit\_memory** — Configures the conditions under which a large memory request is accepted or denied. The following three modes are available:
  - **0** — The kernel performs heuristic memory over commit handling by estimating the amount of memory available and failing requests that are blatantly invalid. Unfortunately, since memory is allocated using a heuristic rather than a precise algorithm, this setting can sometimes allow available memory on the system to be overloaded. This is the default setting.
  - **1** — The kernel performs no memory over commit handling. Under this setting, the potential for memory overload is increased, but so is performance for memory intensive tasks (such as those executed by some scientific software).
  - **2** — The kernel fails any request for memory that would cause the total address space to exceed the sum of the allocated swap space and the percentage of physical RAM specified in **/proc/sys/vm/overcommit\_ratio**. This setting is best for those who desire less risk of memory overcommitment.



#### NOTE

This setting is only recommended for systems with swap areas larger than physical memory.

- **overcommit\_ratio** — Specifies the percentage of physical RAM considered when **/proc/sys/vm/overcommit\_memory** is set to **2**. The default value is **50**.
- **page-cluster** — Sets the number of pages read in a single attempt. The default value of **3**, which actually relates to 16 pages, is appropriate for most systems.
- **swappiness** — Determines how much a machine should swap. The higher the value, the more swapping occurs. The default value, as a percentage, is set to **60**.

All kernel-based documentation can be found in the following locally installed location:

**/usr/share/doc/kernel-doc-kernel\_version/Documentation/**, which contains additional information.

### E.3.10. /proc/sysvipc/

This directory contains information about System V IPC resources. The files in this directory relate to System V IPC calls for messages (**msg**), semaphores (**sem**), and shared memory (**shm**).

### E.3.11. /proc/tty/



This directory contains information about the available and currently used *tty devices* on the system. Originally called *teletype devices*, any character-based data terminals are called tty devices.

In Linux, there are three different kinds of tty devices. *Serial devices* are used with serial connections, such as over a modem or using a serial cable. *Virtual terminals* create the common console connection, such as the virtual consoles available when pressing **Alt+<F-key>** at the system console. *Pseudo terminals* create a two-way communication that is used by some higher level applications, such as XFree86. The **drivers** file is a list of the current tty devices in use, as in the following example:

```
serial          /dev/cua        5  64-127  serial:callout
serial          /dev/ttyS       4  64-127  serial
pty_slave      /dev/pts       136  0-255  pty:slave
pty_master     /dev/ptm       128  0-255  pty:master
pty_slave      /dev/tty        3   0-255  pty:slave
pty_master     /dev/pty        2   0-255  pty:master
/dev/vc/0      /dev/vc/0       4     0  system:vtmaster
/dev/ptmx      /dev/ptmx       5     2  system
/dev/console   /dev/console    5     1  system:console
/dev/tty       /dev/tty        5     0  system:/dev/tty
unknown       /dev/vc/%d      4   1-63  console
```

The **/proc/tty/driver/serial** file lists the usage statistics and status of each of the serial tty lines.

In order for tty devices to be used as network devices, the Linux kernel enforces *line discipline* on the device. This allows the driver to place a specific type of header with every block of data transmitted over the device, making it possible for the remote end of the connection to treat a block of data as just one in a stream of data blocks. SLIP and PPP are common line disciplines, and each are commonly used to connect systems to one another over a serial link.

### E.3.12. /proc/PID/

Out of Memory (OOM) refers to a computing state where all available memory, including swap space, has been allocated. When this situation occurs, it will cause the system to panic and stop functioning as expected. There is a switch that controls OOM behavior in **/proc/sys/vm/panic\_on\_oom**. When set to **1** the kernel will panic on OOM. A setting of **0** instructs the kernel to call a function named **oom\_killer** on an OOM. Usually, **oom\_killer** can kill rogue processes and the system will survive.

The easiest way to change this is to echo the new value to **/proc/sys/vm/panic\_on\_oom**.

```
# cat /proc/sys/vm/panic_on_oom
1

# echo 0 > /proc/sys/vm/panic_on_oom

# cat /proc/sys/vm/panic_on_oom
0
```

It is also possible to prioritize which processes get killed by adjusting the **oom\_killer** score. In **/proc/PID/** there are two tools labeled **oom\_adj** and **oom\_score**. Valid scores for **oom\_adj** are in the range -16 to +15. To see the current **oom\_killer** score, view the **oom\_score** for the process. **oom\_killer** will kill processes with the highest scores first.

This example adjusts the **oom\_score** of a process with a *PID* of 12465 to make it less likely that **oom\_killer** will kill it.

```
# cat /proc/12465/oom_score
79872

# echo -5 > /proc/12465/oom_adj

# cat /proc/12465/oom_score
78
```

There is also a special value of -17, which disables **oom\_killer** for that process. In the example below, **oom\_score** returns a value of 0, indicating that this process would not be killed.

```
# cat /proc/12465/oom_score
78

# echo -17 > /proc/12465/oom_adj

# cat /proc/12465/oom_score
0
```

A function called **badness()** is used to determine the actual score for each process. This is done by adding up 'points' for each examined process. The process scoring is done in the following way:

1. The basis of each process's score is its memory size.
2. The memory size of any of the process's children (not including a kernel thread) is also added to the score
3. The process's score is increased for 'niced' processes and decreased for long running processes.
4. Processes with the **CAP\_SYS\_ADMIN** and **CAP\_SYS\_RAWIO** capabilities have their scores reduced.
5. The final score is then bitshifted by the value saved in the **oom\_adj** file.

Thus, a process with the highest **oom\_score** value will most probably be a non-privileged, recently started process that, along with its children, uses a large amount of memory, has been 'niced', and handles no raw I/O.

## E.4. USING THE SYSCTL COMMAND

The **/sbin/sysctl** command is used to view, set, and automate kernel settings in the **/proc/sys/** directory.

For a quick overview of all settings configurable in the **/proc/sys/** directory, type the **/sbin/sysctl -a** command as root. This creates a large, comprehensive list, a small portion of which looks something like the following:

```
net.ipv4.route.min_pmtu = 552
kernel.sysrq = 0
kernel.sem = 250      32000      32      128
```

This is the same information seen if each of the files were viewed individually. The only difference is the file location. For example, the **/proc/sys/net/ipv4/route/min\_pmtu** file is listed as

`net.ipv4.route.min_pmtu`, with the directory slashes replaced by dots and the `proc.sys` portion assumed.

The `sysctl` command can be used in place of `echo` to assign values to writable files in the `/proc/sys/` directory. For example, instead of using the command

```
echo 1 > /proc/sys/kernel/sysrq
```

use the equivalent `sysctl` command as follows:

```
sysctl -w kernel.sysrq="1"  
kernel.sysrq = 1
```

While quickly setting single values like this in `/proc/sys/` is helpful during testing, this method does not work as well on a production system as special settings within `/proc/sys/` are lost when the machine is rebooted. To preserve custom settings, add them to the `/etc/sysctl.conf` file.

The `/etc/sysctl.conf` file is installed by the `initscripts` package to override some kernel default values and therefore only contains a few of the possible parameters. Use the `sysctl -a` command to list the parameters in the `sysctl` key format. See the `/usr/share/doc/kernel-doc-kernel_version/Documentation/networking/ip-sysctl.txt` file for more information on the possible settings.

Each time the system boots, the `init` program runs the `/etc/rc.d/rc.sysinit` script. This script contains a command to execute `sysctl` using `/etc/sysctl.conf` to determine the values passed to the kernel. Any values added to `/etc/sysctl.conf` therefore take effect each time the system boots. Note that modules loaded after `sysctl` has parsed this file might override the settings.

## E.5. ADDITIONAL RESOURCES

Below are additional sources of information about the `proc` file system.

### Installable Documentation

- `/usr/share/doc/kernel-doc-kernel_version/Documentation/` — This directory, which is provided by the `kernel-doc` package, contains documentation about the `proc` file system. Before accessing the kernel documentation, you must run the following command as root:

```
~]# yum install kernel-doc
```

- `/usr/share/doc/kernel-doc-kernel_version/Documentation/filesystems/proc.txt` — Contains assorted, but limited, information about all aspects of the `/proc/` directory.
- `/usr/share/doc/kernel-doc-kernel_version/Documentation/sysrq.txt` — An overview of System Request Key options.
- `/usr/share/doc/kernel-doc-kernel_version/Documentation/sysctl/` — A directory containing a variety of `sysctl` tips, including modifying values that concern the kernel (`kernel.txt`), accessing file systems (`fs.txt`), and virtual memory use (`vm.txt`).

- `/usr/share/doc/kernel-doc-kernel_version/Documentation/networking/ip-sysctl.txt` — A detailed overview of IP networking options.

## APPENDIX F. REVISION HISTORY

<b>Revision 9-3</b> Version for 6.9 GA publication.	<b>Wed Mar 15 2017</b>	<b>Mirek Jahoda</b>
<b>Revision 8-3</b> The <code>/proc/meminfo</code> appendix section updated; made minor improvements.	<b>Mon May 30 2016</b>	<b>Maxim Svistunov</b>
<b>Revision 8-2</b> Added <i>Relax-and-Recover (ReaR)</i> ; made minor improvements.	<b>Wed May 25 2016</b>	<b>Maxim Svistunov</b>
<b>Revision 8-1</b> Red Hat Enterprise Linux 6.8 GA release of the Deployment Guide.	<b>Thu May 10 2016</b>	<b>Maxim Svistunov</b>
<b>Revision 7-1</b> Red Hat Enterprise Linux 6.7 GA release of the Deployment Guide.	<b>Tue Jul 14 2015</b>	<b>Barbora Ančincová</b>
<b>Revision 7-0</b> Red Hat Enterprise Linux 6.7 Beta release of the Deployment Guide.	<b>Fri Apr 17 2015</b>	<b>Barbora Ančincová</b>
<b>Revision 6-3</b> Updated <i>TigerVNC</i> , <i>Viewing and Managing Log Files</i> , <i>Registering the System and Managing Subscriptions</i> , and <i>The kdump Crash Recovery Service</i> .	<b>Thu Apr 2 2015</b>	<b>Barbora Ančincová</b>
<b>Revision 6-2</b> Red Hat Enterprise Linux 6.6 GA release of the Deployment Guide.	<b>Fri Oct 14 2014</b>	<b>Barbora Ančincová</b>
<b>Revision 6-1</b> Updated <i>NetworkManager</i> , <i>Network Interfaces</i> , <i>Configuring Authentication</i> , <i>The kdump Crash Recovery Service</i> , and <i>The proc File System</i> .	<b>Fri Aug 22 2014</b>	<b>Jaromír Hradílek</b>
<b>Revision 6-0</b> Red Hat Enterprise Linux 6.6 Beta release of the Deployment Guide.	<b>Mon Aug 11 2014</b>	<b>Jaromír Hradílek</b>
<b>Revision 5-1</b> Red Hat Enterprise Linux 6.5 GA release of the Deployment Guide.	<b>Thu Nov 21 2013</b>	<b>Jaromír Hradílek</b>
<b>Revision 5-0</b> Red Hat Enterprise Linux 6.5 Beta release of the Deployment Guide.	<b>Thu Oct 3 2013</b>	<b>Jaromír Hradílek</b>
<b>Revision 4-1</b> Red Hat Enterprise Linux 6.4 GA release of the Deployment Guide.	<b>Thu Feb 21 2013</b>	<b>Jaromír Hradílek</b>
<b>Revision 4-0</b> Red Hat Enterprise Linux 6.4 Beta release of the Deployment Guide.	<b>Thu Dec 6 2012</b>	<b>Jaromír Hradílek</b>
<b>Revision 3-1</b> Red Hat Enterprise Linux 6.3 GA release of the Deployment Guide.	<b>Wed Jun 20 2012</b>	<b>Jaromír Hradílek</b>
<b>Revision 3-0</b> Red Hat Enterprise Linux 6.3 Beta release of the Deployment Guide.	<b>Tue Apr 24 2012</b>	<b>Jaromír Hradílek</b>
<b>Revision 2-1</b> Red Hat Enterprise Linux 6.2 GA release of the Deployment Guide.	<b>Tue Dec 6 2011</b>	<b>Jaromír Hradílek</b>
<b>Revision 2-0</b>	<b>Mon Oct 3 2011</b>	<b>Jaromír Hradílek</b>

Red Hat Enterprise Linux 6.2 Beta release of the Deployment Guide.

**Revision 1-1****Wed May 19 2011****Jaromír Hradílek**

Red Hat Enterprise Linux 6.1 GA release of the Deployment Guide.

**Revision 1-0****Tue Mar 22 2011****Jaromír Hradílek**

Red Hat Enterprise Linux 6.1 Beta release of the Deployment Guide.

**Revision 0-1****Tue Nov 9 2010****Douglas Silas**

Red Hat Enterprise Linux 6.0 GA release of the Deployment Guide.

**Revision 0-0****Mon Nov 16 2009****Douglas Silas**

Initialization of the Red Hat Enterprise Linux 6 Deployment Guide.

# INDEX

## Symbols

`.fetchmailrc`, [Fetchmail Configuration Options](#)

server options, [Server Options](#)

user options, [User Options](#)

`.htaccess` , [Common httpd.conf Directives](#)

(see also [Apache HTTP Server](#) )

`.htpasswd` , [Common httpd.conf Directives](#)

(see also [Apache HTTP Server](#) )

`.procmailrc`, [Procmail Configuration](#)

`/dev/oprofile/`, [Understanding /dev/oprofile/](#)

`/etc/named.conf` (see [BIND](#))

`/etc/sysconfig/` directory (see [sysconfig directory](#))

`/etc/sysconfig/dhcpd`, [Starting and Stopping the Server](#)

`/proc/` directory (see [proc file system](#))

`/var/spool/anacron` , [Configuring Anacron Jobs](#)

`/var/spool/cron` , [Configuring Cron Jobs](#)

(see [OProfile](#))

## A

### Access Control

configuring in SSSD, [Creating Domains: Access Control](#)

SSSD rules, [Creating Domains: Access Control](#)

`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

installable documentation, [Additional Resources](#)

installed documentation, [Additional Resources](#)

useful websites, [Additional Resources](#)

checking configuration, [Editing the Configuration Files](#)

checking status, [Verifying the Service Status](#)

#### directives

`<Directory>` , [Common httpd.conf Directives](#)

`<IfDefine>` , [Common httpd.conf Directives](#)

---

[<IfModule>](#) , [Common httpd.conf Directives](#)  
[<Location>](#) , [Common httpd.conf Directives](#)  
[<Proxy>](#) , [Common httpd.conf Directives](#)  
[<VirtualHost>](#) , [Common httpd.conf Directives](#)  
[AccessFileName](#) , [Common httpd.conf Directives](#)  
[Action](#) , [Common httpd.conf Directives](#)  
[AddDescription](#) , [Common httpd.conf Directives](#)  
[AddEncoding](#) , [Common httpd.conf Directives](#)  
[AddHandler](#) , [Common httpd.conf Directives](#)  
[AddIcon](#) , [Common httpd.conf Directives](#)  
[AddIconByEncoding](#) , [Common httpd.conf Directives](#)  
[AddIconByType](#) , [Common httpd.conf Directives](#)  
[AddLanguage](#) , [Common httpd.conf Directives](#)  
[AddType](#) , [Common httpd.conf Directives](#)  
[Alias](#) , [Common httpd.conf Directives](#)  
[Allow](#) , [Common httpd.conf Directives](#)  
[AllowOverride](#) , [Common httpd.conf Directives](#)  
[BrowserMatch](#) , [Common httpd.conf Directives](#)  
[CacheDefaultExpire](#) , [Common httpd.conf Directives](#)  
[CacheDisable](#) , [Common httpd.conf Directives](#)  
[CacheEnable](#) , [Common httpd.conf Directives](#)  
[CacheLastModifiedFactor](#) , [Common httpd.conf Directives](#)  
[CacheMaxExpire](#) , [Common httpd.conf Directives](#)  
[CacheNegotiatedDocs](#) , [Common httpd.conf Directives](#)  
[CacheRoot](#) , [Common httpd.conf Directives](#)  
[CustomLog](#) , [Common httpd.conf Directives](#)  
[DefaultIcon](#) , [Common httpd.conf Directives](#)  
[DefaultType](#) , [Common httpd.conf Directives](#)  
[Deny](#) , [Common httpd.conf Directives](#)  
[DirectoryIndex](#) , [Common httpd.conf Directives](#)  
[DocumentRoot](#) , [Common httpd.conf Directives](#)  
[ErrorDocument](#) , [Common httpd.conf Directives](#)  
[ErrorLog](#) , [Common httpd.conf Directives](#)  
[ExtendedStatus](#) , [Common httpd.conf Directives](#)  
[Group](#) , [Common httpd.conf Directives](#)  
[HeaderName](#) , [Common httpd.conf Directives](#)  
[HostnameLookups](#) , [Common httpd.conf Directives](#)  
[Include](#) , [Common httpd.conf Directives](#)  
[IndexIgnore](#) , [Common httpd.conf Directives](#)  
[IndexOptions](#) , [Common httpd.conf Directives](#)  
[KeepAlive](#) , [Common httpd.conf Directives](#)  
[KeepAliveTimeout](#) , [Common httpd.conf Directives](#)

---



**LanguagePriority** , [Common httpd.conf Directives](#)  
**Listen** , [Common httpd.conf Directives](#)  
**LoadModule** , [Common httpd.conf Directives](#)  
**LogFormat** , [Common httpd.conf Directives](#)  
**LogLevel** , [Common httpd.conf Directives](#)  
**MaxClients** , [Common Multi-Processing Module Directives](#)  
**MaxKeepAliveRequests** , [Common httpd.conf Directives](#)  
**MaxSpareServers** , [Common Multi-Processing Module Directives](#)  
**MaxSpareThreads** , [Common Multi-Processing Module Directives](#)  
**MinSpareServers** , [Common Multi-Processing Module Directives](#)  
**MinSpareThreads** , [Common Multi-Processing Module Directives](#)  
**NameVirtualHost** , [Common httpd.conf Directives](#)  
**Options** , [Common httpd.conf Directives](#)  
**Order** , [Common httpd.conf Directives](#)  
**PidFile** , [Common httpd.conf Directives](#)  
**ProxyRequests** , [Common httpd.conf Directives](#)  
**ReadmeName** , [Common httpd.conf Directives](#)  
**Redirect** , [Common httpd.conf Directives](#)  
**ScriptAlias** , [Common httpd.conf Directives](#)  
**ServerAdmin** , [Common httpd.conf Directives](#)  
**ServerName** , [Common httpd.conf Directives](#)  
**ServerRoot** , [Common httpd.conf Directives](#)  
**ServerSignature** , [Common httpd.conf Directives](#)  
**ServerTokens** , [Common httpd.conf Directives](#)  
**SetEnvIf** , [Common ssl.conf Directives](#)  
**StartServers** , [Common Multi-Processing Module Directives](#)  
**SuexecUserGroup** , [Common httpd.conf Directives](#)  
**ThreadsPerChild** , [Common Multi-Processing Module Directives](#)  
**Timeout** , [Common httpd.conf Directives](#)  
**TypesConfig** , [Common httpd.conf Directives](#)  
**UseCanonicalName** , [Common httpd.conf Directives](#)  
**User** , [Common httpd.conf Directives](#)  
**UserDir** , [Common httpd.conf Directives](#)

#### directories

**/etc/httpd/** , [Common httpd.conf Directives](#)  
**/etc/httpd/conf.d/** , [Editing the Configuration Files](#), [Common httpd.conf Directives](#)  
**/usr/lib/httpd/modules/** , [Common httpd.conf Directives](#), [Working with Modules](#)  
**/usr/lib64/httpd/modules/** , [Common httpd.conf Directives](#), [Working with Modules](#)  
**/var/cache/mod\_proxy/** , [Common httpd.conf Directives](#)  
**/var/www/cgi-bin/** , [Common httpd.conf Directives](#)  
**/var/www/html/** , [Common httpd.conf Directives](#)

---

[/var/www/icons/](#) , [Common httpd.conf Directives](#)

[~/public\\_html/](#) , [Common httpd.conf Directives](#)

## files

[.htaccess](#) , [Common httpd.conf Directives](#)

[.htpasswd](#) , [Common httpd.conf Directives](#)

[/etc/httpd/conf.d/nss.conf](#) , [Enabling the mod\\_nss Module](#)

[/etc/httpd/conf.d/ssl.conf](#) , [Common ssl.conf Directives](#) , [Enabling the mod\\_ssl Module](#)

[/etc/httpd/conf/httpd.conf](#) , [Editing the Configuration Files](#) , [Common httpd.conf Directives](#) , [Common Multi-Processing Module Directives](#)

[/etc/httpd/logs/access\\_log](#) , [Common httpd.conf Directives](#)

[/etc/httpd/logs/error\\_log](#) , [Common httpd.conf Directives](#)

[/etc/httpd/run/httpd.pid](#) , [Common httpd.conf Directives](#)

[/etc/mime.types](#) , [Common httpd.conf Directives](#)

## modules

developing, [Writing a Module](#)

loading, [Loading a Module](#)

[mod\\_asis](#), [Notable Changes](#)

[mod\\_cache](#), [New Features](#)

[mod\\_cern\\_meta](#), [Notable Changes](#)

[mod\\_disk\\_cache](#), [New Features](#)

[mod\\_ext\\_filter](#), [Notable Changes](#)

[mod\\_proxy\\_balancer](#), [New Features](#)

[mod\\_rewrite](#) , [Common httpd.conf Directives](#)

[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 2.2

changes, [Notable Changes](#)

features, [New Features](#)

updating from version 2.0, [Updating the Configuration](#)

virtual host, [Setting Up Virtual Hosts](#)

at , [At and Batch](#)

additional resources, [Additional Resources](#)

authconfig (see [Authentication Configuration Tool](#))

commands, [Configuring Authentication from the Command Line](#)

authentication

[Authentication Configuration Tool](#), [Configuring System Authentication](#)

using fingerprint support, [Using Fingerprint Authentication](#)

using smart card authentication, [Enabling Smart Card Authentication](#)

[Authentication Configuration Tool](#)

and Kerberos authentication, [Using Kerberos with LDAP or NIS Authentication](#)

and LDAP, [Configuring LDAP Authentication](#)

and NIS, [Configuring NIS Authentication](#)

and Winbind, [Configuring Winbind Authentication](#)

and Winbind authentication, [Configuring Winbind Authentication](#)

authoritative nameserver (see [BIND](#))

Automated Tasks, [Automating System Tasks](#)

## B

batch , [At and Batch](#)

additional resources, [Additional Resources](#)

Berkeley Internet Name Domain (see [BIND](#))

[BIND](#)

additional resources

installed documentation, [Installed Documentation](#)

related books, [Related Books](#)

useful websites, [Useful Websites](#)

common mistakes, [Common Mistakes to Avoid](#)

configuration

acl statement, [Common Statement Types](#)

comment tags, [Comment Tags](#)

controls statement, [Other Statement Types](#)

include statement, [Common Statement Types](#)

key statement, [Other Statement Types](#)

logging statement, [Other Statement Types](#)

options statement, [Common Statement Types](#)

server statement, [Other Statement Types](#)

trusted-keys statement, [Other Statement Types](#)

view statement, [Other Statement Types](#)  
zone statement, [Common Statement Types](#)

## directories

/etc/named/ , [Configuring the named Service](#)  
/var/named/ , [Editing Zone Files](#)  
/var/named/data/ , [Editing Zone Files](#)  
/var/named/dynamic/ , [Editing Zone Files](#)  
/var/named/slaves/ , [Editing Zone Files](#)

## features

Automatic Zone Transfer (AXFR), [Incremental Zone Transfers \(IXFR\)](#)  
DNS Security Extensions (DNSSEC), [DNS Security Extensions \(DNSSEC\)](#)  
Incremental Zone Transfer (IXFR), [Incremental Zone Transfers \(IXFR\)](#)  
Internet Protocol version 6 (IPv6), [Internet Protocol version 6 \(IPv6\)](#)  
multiple views, [Multiple Views](#)  
Transaction SIGNature (TSIG), [Transaction SIGNatures \(TSIG\)](#)

## files

/etc/named.conf , [Configuring the named Service](#), [Configuring the Utility](#)  
/etc/rndc.conf , [Configuring the Utility](#)  
/etc/rndc.key , [Configuring the Utility](#)

resource record, [Nameserver Zones](#)

## types

authoritative nameserver, [Nameserver Types](#)  
primary (master) nameserver, [Nameserver Zones](#), [Nameserver Types](#)  
recursive nameserver, [Nameserver Types](#)  
secondary (slave) nameserver, [Nameserver Zones](#), [Nameserver Types](#)

## utilities

dig, [BIND as a Nameserver](#), [Using the dig Utility](#), [DNS Security Extensions \(DNSSEC\)](#)  
named, [BIND as a Nameserver](#), [Configuring the named Service](#)  
rndc, [BIND as a Nameserver](#), [Using the rndc Utility](#)

## zones

\$INCLUDE directive, [Common Directives](#)  
\$ORIGIN directive, [Common Directives](#)  
\$TTL directive, [Common Directives](#)  
A (Address) resource record, [Common Resource Records](#)  
CNAME (Canonical Name) resource record, [Common Resource Records](#)  
comment tags, [Comment Tags](#)  
description, [Nameserver Zones](#)  
example usage, [A Simple Zone File](#), [A Reverse Name Resolution Zone File](#)

MX (Mail Exchange) resource record, [Common Resource Records](#)  
NS (Nameserver) resource record, [Common Resource Records](#)  
PTR (Pointer) resource record, [Common Resource Records](#)  
SOA (Start of Authority) resource record, [Common Resource Records](#)

blkid, [Using the blkid Command](#)

block devices, [/proc/devices](#)

(see also [/proc/devices](#))

definition of, [/proc/devices](#)

bonding (see channel bonding)

boot loader

verifying, [Verifying the Boot Loader](#)

boot media, [Preparing to Upgrade](#)

## C

ch-email .fetchmailrc

global options, [Global Options](#)

channel bonding

configuration, [Using Channel Bonding](#)

description, [Using Channel Bonding](#)

interface

configuration of, [Channel Bonding Interfaces](#)

parameters to bonded interfaces, [Bonding Module Directives](#)

channel bonding interface (see kernel module)

character devices, [/proc/devices](#)

(see also [/proc/devices](#))

definition of, [/proc/devices](#)

chkconfig (see services configuration)

Configuration File Changes, [Preserving Configuration File Changes](#)

CPU usage, [Viewing CPU Usage](#)

crash

analyzing the dump

message buffer, [Displaying the Message Buffer](#)

open files, [Displaying Open Files](#)

processes, [Displaying a Process Status](#)

stack trace, [Displaying a Backtrace](#)

virtual memory, [Displaying Virtual Memory Information](#)

opening the dump image, [Running the crash Utility](#)  
system requirements, [Analyzing the Core Dump](#)

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

date (see [date configuration](#))

date configuration

date, [Date and Time Setup](#)

system-config-date, [Date and Time Properties](#)

default gateway, [Static Routes and the Default Gateway](#)

deleting cache files

in SSSD, [Managing the SSSD Cache](#)

Denial of Service attack, [/proc/sys/net/](#)

(see also [/proc/sys/net/](#) directory)

definition of, [/proc/sys/net/](#)

desktop environments (see [X](#))

df, [Using the df Command](#)

DHCP, [DHCP Servers](#)

additional resources, [Additional Resources](#)

client configuration, [Configuring a DHCPv4 Client](#)

command-line options, [Starting and Stopping the Server](#)

connecting to, [Configuring a DHCPv4 Client](#)

dhcpd.conf, [Configuration File](#)

dhcpd.leases, [Starting and Stopping the Server](#)

dhcpd6.conf, [DHCP for IPv6 \(DHCPv6\)](#)

DHCPv6, [DHCP for IPv6 \(DHCPv6\)](#)

dhcrelay, [DHCP Relay Agent](#)

global parameters, [Configuration File](#)

group, [Configuration File](#)

options, [Configuration File](#)

reasons for using, [Why Use DHCP?](#)

Relay Agent, [DHCP Relay Agent](#)

shared-network, [Configuration File](#)

starting the server, [Starting and Stopping the Server](#)  
stopping the server, [Starting and Stopping the Server](#)  
subnet, [Configuration File](#)

dhcpd.conf, [Configuration File](#)

dhcpd.leases, [Starting and Stopping the Server](#)

DHCPv4

server configuration, [Configuring a DHCPv4 Server](#)

dhcrelay, [DHCP Relay Agent](#)

dig (see BIND)

directory server (see OpenLDAP)

display managers (see X)

DNS

definition, [DNS Servers](#)

(see also BIND)

documentation

finding installed, [Practical and Common Examples of RPM Usage](#)

DoS attack (see Denial of Service attack)

downgrade

and SSSD, [Downgrading SSSD](#)

drivers (see kernel module)

DSA keys

generating, [Generating Key Pairs](#)

du, [Using the du Command](#)

Dynamic Host Configuration Protocol (see DHCP)

**E**

email

additional resources, [Additional Resources](#)

installed documentation, [Installed Documentation](#)

online documentation, [Online Documentation](#)

related books, [Related Books](#)

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](#)

epoch, [/proc/stat](#)

(see also [/proc/stat](#))

definition of, [/proc/stat](#)

Ethernet (see network)

Ethtool

command

devname , [Ethtool](#)

option

--advertise , [Ethtool](#)

--autoneg , [Ethtool](#)

--duplex , [Ethtool](#)

--features , [Ethtool](#)

--identify , [Ethtool](#)

--msglvl , [Ethtool](#)

--phyad , [Ethtool](#)

--port , [Ethtool](#)

--show-features , [Ethtool](#)

--show-time-stamping , [Ethtool](#)

--sopass , [Ethtool](#)

--speed , [Ethtool](#)

--statistics , [Ethtool](#)

--test , [Ethtool](#)



--wol , [Ethtool](#)

--xcvr , [Ethtool](#)

## exec-shield

enabling, [/proc/sys/kernel/](#)

introducing, [/proc/sys/kernel/](#)

## execution domains, [/proc/execd domains](#)

(see also [/proc/execd domains](#))

definition of, [/proc/execd domains](#)

## 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 system

virtual (see [proc file system](#))

### file systems, [Viewing Block Devices and File Systems](#)

### files, [proc file system](#)

changing, [Changing Virtual Files, Using the sysctl Command](#)

viewing, [Viewing Virtual Files, Using the sysctl Command](#)

### findmnt, [Using the findmnt Command](#)

### findsmb, [Connecting to a Samba Share](#)

### findsmb program, [Samba Distribution Programs](#)

### FQDN (see [fully qualified domain name](#))

### frame buffer device, [/proc/fb](#)

(see also [/proc/fb](#))

### 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](#)

fully qualified domain name, [Nameserver Zones](#)

## G

gamin, [Monitoring Files and Directories with gamin](#)

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 a Package's Signature](#)

group configuration

modifying group properties, [Modifying Group Properties](#)

groups

additional resources, [Additional Resources](#)

installed documentation, [Installed Documentation](#)

GRUB boot loader

configuration file, [Configuring the GRUB Boot Loader](#)

configuring, [Configuring the GRUB Boot Loader](#)

## H

hardware

viewing, [Viewing Hardware Information](#)

HTTP server (see [Apache HTTP Server](#))

httpd (see [Apache HTTP Server](#))

hugepages

configuration of, [/proc/sys/vm/](#)

## I

ifdown, [Interface Control Scripts](#)

**ifup**, [Interface Control Scripts](#)

**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 package groups**

installing package groups with PackageKit, [Installing and Removing Package Groups](#)

**installing the kernel**, [Manually Upgrading the Kernel](#)

**K**

**KDE**, [Desktop Environments](#)

(see also X)

**kdump**

**additional resources**

installed documents, [Additional Resources](#)

manual pages, [Additional Resources](#)

websites, [Additional Resources](#)

**analyzing the dump** (see crash)

**configuring the service**

default action, [Using the Kernel Dump Configuration Utility, Configuring kdump on the Command Line](#)

dump image compression, [Using the Kernel Dump Configuration Utility, Configuring kdump on the Command Line](#)

filtering level, [Using the Kernel Dump Configuration Utility, Configuring kdump on the Command Line](#)

initial RAM disk, [Using the Kernel Dump Configuration Utility, Configuring kdump on the Command Line](#)

kernel image, [Using the Kernel Dump Configuration Utility, Configuring kdump on the Command Line](#)

kernel options, [Using the Kernel Dump Configuration Utility, Configuring kdump on the Command Line](#)

memory usage, [Configuring kdump at First Boot, Using the Kernel Dump Configuration Utility, Configuring kdump on the Command Line](#)

supported targets, [Using the Kernel Dump Configuration Utility](#), [Configuring kdump on the Command Line](#)

target location, [Using the Kernel Dump Configuration Utility](#), [Configuring kdump on the Command Line](#)

enabling the service, [Configuring kdump at First Boot](#), [Using the Kernel Dump Configuration Utility](#), [Configuring kdump on the Command Line](#)

fadump, [Using fadump on IBM PowerPC hardware](#)

installing, [Installing the kdump Service](#)

running the service, [Configuring kdump on the Command Line](#)

system requirements, [Configuring the kdump Service](#)

testing the configuration, [Testing the Configuration](#)

## 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](#)

Red Hat network, [Downloading the Upgraded Kernel](#)

Security Errata, [Downloading the Upgraded Kernel](#)

## upgrading

preparing, [Preparing to Upgrade](#)

working boot media, [Preparing to Upgrade](#)

upgrading the kernel, [Manually Upgrading the Kernel](#)

## Kernel Dump Configuration (see kdump)

### kernel module

bonding module, [Using Channel Bonding](#)

description, [Using Channel Bonding](#)

parameters to bonded interfaces, [Bonding Module Directives](#)

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

bonding module parameters, [Bonding Module Directives](#)

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-doc

documentation files, [Overview of Kernel Packages](#)

##### kernel-firmware

firmware files, [Overview of Kernel Packages](#)

##### kernel-headers

C header files files, [Overview of Kernel Packages](#)

##### perf

firmware files, [Overview of Kernel Packages](#)

#### kernel upgrading

preparing, [Preparing to Upgrade](#)

#### keyboard configuration, [Keyboard Configuration](#)

Keyboard Indicator applet, [Adding the Keyboard Layout Indicator](#)

Keyboard Preferences utility, [Changing the Keyboard Layout](#)

layout, [Changing the Keyboard Layout](#)

typing break, [Setting Up a Typing Break](#)

---

Keyboard Indicator (see keyboard configuration)  
Keyboard Preferences (see keyboard configuration)  
kwin, [Window Managers](#)  
(see also X)

## L

LDAP (see OpenLDAP)  
Log File Viewer, [Managing Log Files in a Graphical Environment](#)  
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)

lspci, [Using the lspci Command](#), [/proc/bus/pci](#)  
lspcmcia, [Using the lspcmcia Command](#)  
lsusb, [Using the lsusb 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)

**Multihomed DHCP**

host configuration, [Host Configuration](#)

server configuration, [Configuring a Multihomed DHCP Server](#)

**mwm**, [Window Managers](#)

(see also X)

## N

**named** (see BIND)

**nameserver** (see DNS)

**net program**, [Samba Distribution Programs](#)

**network**

additional resources, [Additional Resources](#)

**bridge**

bridging, [Network Bridge](#)

**commands**

/sbin/ifdown, [Interface Control Scripts](#)

/sbin/ifup, [Interface Control Scripts](#)

/sbin/service network, [Interface Control Scripts](#)

**configuration**, [Interface Configuration Files](#)

**configuration files**, [Network Configuration Files](#)

**functions**, [Network Function Files](#)

**interface configuration files**, [Interface Configuration Files](#)

**interfaces**

802.1Q, [Setting Up 802.1Q VLAN Tagging](#)

alias, [Alias and Clone Files](#)

channel bonding, [Channel Bonding Interfaces](#)

clone, [Alias and Clone Files](#)

dialup, [Dialup Interfaces](#)

Ethernet, [Ethernet Interfaces](#)

---

ethtool, [Ethtool](#)

VLAN, [Setting Up 802.1Q VLAN Tagging](#)

scripts, [Network Interfaces](#)

Network Time Protocol (see [NTP](#))

NIC

binding into single channel, [Using Channel Bonding](#)

nmblookup program, [Samba Distribution Programs](#)

NSCD

and SSSD, [Using NSCD with SSSD](#)

NTP

configuring, [Network Time Protocol Properties](#), [Network Time Protocol Setup](#)

ntpd, [Network Time Protocol Properties](#), [Network Time Protocol Setup](#)

ntpdate, [Network Time Protocol Setup](#)

ntpd (see [NTP](#))

ntpdate (see [NTP](#))

ntsysv (see [services configuration](#))

O

opannotate (see [OProfile](#))

opcontrol (see [OProfile](#))

OpenLDAP

checking status, [Checking 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](#)

olcDisallows, [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={2}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](#))

additional resources, [Additional Resources](#)

client, [OpenSSH Clients](#)

**scp**, [Using the scp Utility](#)

**sftp**, [Using the sftp Utility](#)

**ssh**, [Using the ssh Utility](#)

#### DSA keys

generating, [Generating Key Pairs](#)

#### RSA keys

generating, [Generating Key Pairs](#)

#### RSA Version 1 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

DSA, [Generating Key Pairs](#)  
RSA, [Generating Key Pairs](#)  
RSA Version 1, [Generating Key Pairs](#)

using key-based authentication, [Using Key-Based Authentication](#)

## OpenSSL

additional resources, [Additional Resources](#)  
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](#)

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](#)

`--no-vmlinux`, [Specifying the Kernel](#)

`--start`, [Starting and Stopping OProfile](#)

`--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](#)

log file, [Starting and Stopping OProfile](#)

overview of tools, [Overview of Tools](#)

reading data, [Analyzing the Data](#)

saving data, [Saving Data](#)

starting, [Starting and Stopping OProfile](#)

SystemTap, [OProfile and SystemTap](#)

unit mask, [Unit Masks](#)

oprofiled (see [OProfile](#))

oprof\_start, [Graphical Interface](#)

OS/400 boot loader

configuration file, [Configuring the OS/400 Boot Loader](#)

configuring, [Configuring the OS/400 Boot Loader](#)

## P

package

kernel RPM, [Manually Upgrading the Kernel](#)

PackageKit, [PackageKit](#)

adding and removing, [Using Add/Remove Software](#)

architecture, [PackageKit Architecture](#)

installing and removing package groups, [Installing and Removing Package Groups](#)

installing packages, [PackageKit](#)

managing packages, [PackageKit](#)

PolicyKit

authentication, [Updating Packages with Software Update](#)

uninstalling packages, [PackageKit](#)

updating packages, [PackageKit](#)

viewing packages, [PackageKit](#)

viewing transaction log, [Viewing the Transaction Log](#)

packages

adding and removing with PackageKit, [Using Add/Remove Software](#)

dependencies, [Unresolved Dependency](#)

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](#)

extra packages for Enterprise Linux (EPEL), [Finding RPM Packages](#)

filtering with PackageKit, [Finding Packages with Filters](#)

Development, [Finding Packages with Filters](#)

---

- Free, [Finding Packages with Filters](#)
- Hide subpackages, [Finding Packages with Filters](#)
- Installed, [Finding Packages with Filters](#)
- No filter, [Finding Packages with Filters](#)
- Only available, [Finding Packages with Filters](#)
- Only development, [Finding Packages with Filters](#)
- Only end user files, [Finding Packages with Filters](#)
- Only graphical, [Finding Packages with Filters](#)
- Only installed, [Finding Packages with Filters](#)
- Only native packages, [Finding Packages with Filters](#)
- Only newest packages, [Finding Packages with Filters](#)

- filtering with PackageKit for packages, [Finding Packages with Filters](#)
- finding deleted files from, [Practical and Common Examples of RPM Usage](#)
- finding RPM packages, [Finding RPM Packages](#)
- initial RPM repositories, [Finding RPM Packages](#)
- installing a package group with Yum, [Installing Packages](#)
- installing and removing package groups, [Installing and Removing Package Groups](#)
- installing packages with PackageKit, [PackageKit](#), [Installing and Removing Packages \(and Dependencies\)](#)
  - dependencies, [Installing and Removing Packages \(and Dependencies\)](#)

- installing RPM, [Installing and Upgrading](#)
- installing with Yum, [Installing Packages](#)
- iRed Hat Enterprise Linux installation media, [Finding RPM Packages](#)

- kernel
  - for single,multicore and multiprocessor systems, [Overview of Kernel Packages](#)

- kernel-devel
  - kernel headers and makefiles, [Overview of Kernel Packages](#)

- kernel-doc
  - documentation files, [Overview of Kernel Packages](#)

- kernel-firmware
  - firmware files, [Overview of Kernel Packages](#)

- kernel-headers
  - C header files files, [Overview of Kernel Packages](#)

- listing packages with Yum
  - Glob expressions, [Listing Packages](#)
  - yum grouplist, [Listing Packages](#)
  - yum list all, [Listing 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](#)

managing packages with PackageKit, [PackageKit](#)

obtaining list of files, [Practical and Common Examples of RPM Usage](#)

packages and package groups, [Packages and Package Groups](#)

perf

firmware files, [Overview of Kernel Packages](#)

querying uninstalled, [Practical and Common Examples of RPM Usage](#)

removing, [Uninstalling](#)

removing package groups with Yum, [Removing Packages](#)

removing packages with PackageKit, [Installing and Removing Packages \(and Dependencies\)](#)

RPM, [RPM](#)

already installed, [Package Already Installed](#)

configuration file changes, [Configuration File Changes](#)

conflict, [Conflicting Files](#)

failed dependencies, [Unresolved Dependency](#)

freshening, [Freshening](#)

pristine sources, [RPM Design Goals](#)

querying, [Querying](#)

removing, [Uninstalling](#)

source and binary packages, [RPM](#)

tips, [Practical and Common Examples of RPM Usage](#)

uninstalling, [Uninstalling](#)

verifying, [Verifying](#)

searching packages with Yum

yum search, [Searching Packages](#)

setting packages with PackageKit

checking interval, [Updating Packages with Software Update](#)

uninstalling packages with PackageKit, [PackageKit](#)

uninstalling packages with Yum, [Removing Packages](#)

yum remove package\_name, [Removing Packages](#)

updating currently installed packages

available updates, [Updating Packages with Software Update](#)

updating packages with PackageKit, [PackageKit](#)

PolicyKit, [Updating Packages with Software Update](#)

---

Software Update, [Updating Packages with Software Update](#)

upgrading RPM, [Installing and Upgrading](#)

viewing packages with PackageKit, [PackageKit](#)

viewing transaction log, [Viewing the Transaction Log](#)

viewing Yum repositories with PackageKit, [Refreshing Software Sources \(Yum Repositories\)](#)

Yum instead of RPM, [RPM](#)

pdbedit program, [Samba Distribution Programs](#)

PolicyKit, [Updating Packages with Software Update](#)

Postfix, [Postfix](#)

default installation, [The Default Postfix Installation](#)

postfix, [Mail Transport Agent \(MTA\) Configuration](#)

prefdm (see X)

primary nameserver (see BIND)

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)

proc file system

[/proc/buddyinfo](#), [/proc/buddyinfo](#)

[/proc/bus/](#) directory, [/proc/bus/](#)

[/proc/bus/pci](#)

viewing using `lspci`, [/proc/bus/pci](#)

[/proc/cmdline](#), [/proc/cmdline](#)

[/proc/cpuinfo](#), [/proc/cpuinfo](#)

[/proc/crypto](#), [/proc/crypto](#)

[/proc/devices](#)

block devices, [/proc/devices](#)

character devices, [/proc/devices](#)

[/proc/dma](#), [/proc/dma](#)

[/proc/driver/](#) directory, [/proc/driver/](#)

[/proc/execdomains](#), [/proc/execdomains](#)

[/proc/fb](#), [/proc/fb](#)  
[/proc/filesystems](#), [/proc/filesystems](#)  
[/proc/fs/](#) directory, [/proc/fs](#)  
[/proc/interrupts](#), [/proc/interrupts](#)  
[/proc/iomem](#), [/proc/iomem](#)  
[/proc/ioports](#), [/proc/ioports](#)  
[/proc/irq/](#) directory, [/proc/irq/](#)  
[/proc/kcore](#), [/proc/kcore](#)  
[/proc/kmsg](#), [/proc/kmsg](#)  
[/proc/loadavg](#), [/proc/loadavg](#)  
[/proc/locks](#), [/proc/locks](#)  
[/proc/mdstat](#), [/proc/mdstat](#)  
[/proc/meminfo](#), [/proc/meminfo](#)  
[/proc/misc](#), [/proc/misc](#)  
[/proc/modules](#), [/proc/modules](#)  
[/proc/mounts](#), [/proc/mounts](#)  
[/proc/mtrr](#), [/proc/mtrr](#)  
[/proc/net/](#) directory, [/proc/net/](#)  
[/proc/partitions](#), [/proc/partitions](#)  
[/proc/PID/](#) directory, [/proc/PID/](#)  
[/proc/scsi/](#) directory, [/proc/scsi/](#)  
[/proc/self/](#) directory, [/proc/self/](#)  
[/proc/slabinfo](#), [/proc/slabinfo](#)  
[/proc/stat](#), [/proc/stat](#)  
[/proc/swaps](#), [/proc/swaps](#)  
[/proc/sys/](#) directory, [/proc/sys/](#), [Using the sysctl Command](#)  
(see also [sysctl](#))  
[/proc/sys/dev/](#) directory, [/proc/sys/dev/](#)  
[/proc/sys/fs/](#) directory, [/proc/sys/fs/](#)  
[/proc/sys/kernel/](#) directory, [/proc/sys/kernel/](#)  
[/proc/sys/kernel/exec-shield](#), [/proc/sys/kernel/](#)  
[/proc/sys/kernel/sysrq](#) (see [system request key](#))  
[/proc/sys/net/](#) directory, [/proc/sys/net/](#)  
[/proc/sys/vm/](#) directory, [/proc/sys/vm/](#)

[/proc/sysrq-trigger](#), [/proc/sysrq-trigger](#)  
[/proc/sysvipc/](#) directory, [/proc/sysvipc/](#)  
[/proc/tty/](#) directory, [/proc/tty/](#)  
[/proc/uptime](#), [/proc/uptime](#)  
[/proc/version](#), [/proc/version](#)

additional resources, [Additional Resources](#)  
installed documentation, [Additional Resources](#)

---

changing files within, [Changing Virtual Files](#), [/proc/sys/](#), [Using the sysctl Command](#)  
files within, top-level, [Top-level Files within the proc File System](#)  
introduced, [The proc File System](#)  
process directories, [Process Directories](#)  
subdirectories within, [Directories within /proc/](#)  
viewing files within, [Viewing Virtual Files](#)

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](#)

rcp, [Using the scp Utility](#)

ReaR

basic usage, [Basic ReaR Usage](#)

recursive nameserver (see BIND)

Red Hat Support Tool

getting support on the command line, [Accessing Support Using the Red Hat Support Tool](#)

Red Hat Enterprise Linux installation media

installable packages, [Finding RPM Packages](#)

Red Hat Subscription Management

subscription, [Registering the System and Attaching Subscriptions](#)

removing package groups

removing package groups with PackageKit, [Installing and Removing Package Groups](#)

resource record (see BIND)

rmmod, [Unloading a Module](#)



(see also kernel module)

**rndc** (see BIND)

**root nameserver** (see BIND)

**rpcclient** program, [Samba Distribution Programs](#)

**RPM, RPM**

additional resources, [Additional Resources](#)

already installed, [Package Already Installed](#)

basic modes, [Using RPM](#)

checking package signatures, [Checking a Package's Signature](#)

configuration file changes, [Configuration File Changes](#)

conf.rpmsave, [Configuration File Changes](#)

conflicts, [Conflicting Files](#)

dependencies, [Unresolved Dependency](#)

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, [Unresolved Dependency](#)

file conflicts

resolving, [Conflicting Files](#)

file name, [Installing and Upgrading](#)

finding deleted files with, [Practical and Common Examples of RPM Usage](#)

finding RPM packages, [Finding RPM Packages](#)

freshening, [Freshening](#)

GnuPG, [Checking a Package's Signature](#)

installing, [Installing and Upgrading](#)

md5sum, [Checking a Package's Signature](#)

querying, [Querying](#)

querying for file list, [Practical and Common Examples of RPM Usage](#)

querying uninstalled packages, [Practical and Common Examples of RPM Usage](#)

tips, [Practical and Common Examples of RPM Usage](#)

uninstalling, [Uninstalling](#)

upgrading, [Installing and Upgrading](#)

verifying, [Verifying](#)

website, [Useful Websites](#)

**RPM Package Manager** (see RPM)

**RSA keys**

---

generating, [Generating Key Pairs](#)

## RSA Version 1 keys

generating, [Generating Key Pairs](#)

## rsyslog, [Viewing and Managing Log Files](#)

actions, [Actions](#)

configuration, [Basic Configuration of Rsyslog](#)

debugging, [Debugging Rsyslog](#)

filters, [Filters](#)

global directives, [Global Directives](#)

log rotation, [Log Rotation](#)

modules, [Using Rsyslog Modules](#)

new configuration format, [Using the New Configuration Format](#)

queues, [Working with Queues in Rsyslog](#)

rulesets, [Rulesets](#)

templates, [Templates](#)

runlevel (see services configuration)

## S

### Samba (see Samba)

Abilities, [Introduction to Samba](#)

Account Information Databases, [Samba Account Information Databases](#)

Idapsam, [Samba Account Information Databases](#)

Idapsam\_compat, [Samba Account Information Databases](#)

mysqisam, [Samba Account Information Databases](#)

Plain Text, [Samba Account Information Databases](#)

smbpasswd, [Samba Account Information Databases](#)

tdbsam, [Samba Account Information Databases](#)

xmlsam, [Samba Account Information Databases](#)

Additional Resources, [Additional Resources](#)

installed documentation, [Additional Resources](#)

related books, [Additional Resources](#)

useful websites, [Additional Resources](#)

Backward Compatible Database Back Ends, [Samba Account Information Databases](#)

Browsing, [Samba Network Browsing](#)

configuration, [Configuring a Samba Server](#), [Command-Line Configuration](#)

default, [Configuring a Samba Server](#)

CUPS Printing Support, [Samba with CUPS Printing Support](#)

CUPS smb.conf, [Simple smb.conf Settings](#)

**daemon**

**nmbd**, [Samba Daemons and Related Services](#)  
**overview**, [Samba Daemons and Related Services](#)  
**smbd**, [Samba Daemons and Related Services](#)  
**winbindd**, [Samba Daemons and Related Services](#)

**encrypted passwords**, [Encrypted Passwords](#)

**findsmb**, [Connecting to a Samba Share](#)

**graphical configuration**, [Graphical Configuration](#)

**Introduction**, [Introduction to Samba](#)

**Network Browsing**, [Samba Network Browsing](#)

**Domain Browsing**, [Domain Browsing](#)

**WINS**, [WINS \(Windows Internet Name Server\)](#)

**New Database Back Ends**, [Samba Account Information Databases](#)

**Programs**, [Samba Distribution Programs](#)

**findsmb**, [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](#)

**Security Modes**, [Samba Security Modes](#), [User-Level Security](#)

**Active Directory Security Mode**, [User-Level Security](#)

**Domain Security Mode**, [User-Level Security](#)

**Share-Level Security**, [Share-Level Security](#)

**User Level Security**, [User-Level Security](#)

**Server Types**, [Samba Server Types and the smb.conf File](#)

**server types**

**Domain Controller**, [Domain Controller](#)

**Domain Member**, [Domain Member Server](#)

---

Stand Alone, [Stand-alone Server](#)

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, [Connecting to a Samba Share](#)

connecting to with Nautilus, [Connecting to a Samba Share](#)

mounting, [Mounting the Share](#)

smb.conf, [Samba Server Types and the smb.conf File](#)

Active Directory Member Server example, [Domain Member Server](#)

Anonymous Print Server example, [Stand-alone Server](#)

Anonymous Read Only example, [Stand-alone Server](#)

Anonymous Read/Write example, [Stand-alone Server](#)

NT4-style Domain Member example, [Domain Member Server](#)

PDC using Active Directory, [Domain Controller](#)

PDC using tdbsam, [Domain Controller](#)

Secure File and Print Server example, [Stand-alone Server](#)

smbclient, [Connecting to a Samba Share](#)

WINS, [WINS \(Windows Internet Name Server\)](#)

scp (see [OpenSSH](#))

secondary nameserver (see [BIND](#))

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](#)

service (see [services configuration](#))

services configuration, [Services and Daemons](#)

chkconfig, [Using the chkconfig Utility](#)

ntsysv, [Using the ntsysv Utility](#)

runlevel, [Configuring the Default Runlevel](#)

service, [Running Services](#)

system-config-services, [Using the Service Configuration Utility](#)

sftp (see [OpenSSH](#))

slab pools (see [/proc/slabinfo](#))

slapd (see [OpenLDAP](#))

smbcacls program, [Samba Distribution Programs](#)

smbclient, [Connecting to a Samba Share](#)

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](#)

SSL , [Setting Up an SSL Server](#)

(see also [Apache HTTP Server](#) )

SSL server (see [Apache HTTP Server](#) )

SSSD

and NSCD, [Using NSCD with SSSD](#)

configuration file

creating, [Setting up the sssd.conf File](#)

location, [Using a Custom Configuration File](#)

sections, [Creating the sssd.conf File](#)

downgrading, [Downgrading SSSD](#)

identity provider

local, [Creating the sssd.conf File](#)

Kerberos authentication, [Creating Domains: Kerberos Authentication](#)

LDAP domain, [Creating Domains: LDAP](#)

supported LDAP directories, [Creating Domains: LDAP](#)

Microsoft Active Directory domain, [Creating Domains: Active Directory](#), [Configuring Domains: Active Directory as an LDAP Provider \(Alternative\)](#)

proxy domain, [Creating Domains: Proxy](#)

sudo rules

rules stored per host, [Configuring Services: sudo](#)

startx, [Runlevel 3](#) (see [X](#))

(see also [X](#))

static route, [Static Routes and the Default Gateway](#)

stunnel, [Securing Email Client Communications](#)

subscriptions, [Registering the System and Managing Subscriptions](#)

sysconfig directory

[/etc/sysconfig/apm-scripts/](#) directory, [Directories in the /etc/sysconfig/ Directory](#)

[/etc/sysconfig/arpwatch](#), [/etc/sysconfig/arpwatch](#)

[/etc/sysconfig/authconfig](#), [/etc/sysconfig/authconfig](#)

[/etc/sysconfig/autofs](#), [/etc/sysconfig/autofs](#)

[/etc/sysconfig/cbq/](#) directory, [Directories in the /etc/sysconfig/ Directory](#)

[/etc/sysconfig/clock](#), [/etc/sysconfig/clock](#)

[/etc/sysconfig/dhcpd](#), [/etc/sysconfig/dhcpd](#)

[/etc/sysconfig/firstboot](#), [/etc/sysconfig/firstboot](#)

[/etc/sysconfig/init](#), [/etc/sysconfig/init](#)

[/etc/sysconfig/ip6tables-config](#), [/etc/sysconfig/ip6tables-config](#)

[/etc/sysconfig/keyboard](#), [/etc/sysconfig/keyboard](#)

[/etc/sysconfig/ldap](#), [/etc/sysconfig/ldap](#)

[/etc/sysconfig/named](#), [/etc/sysconfig/named](#)

[/etc/sysconfig/network](#), [/etc/sysconfig/network](#)

[/etc/sysconfig/network-scripts/](#) [directory](#), [Network Interfaces](#), [Directories in the /etc/sysconfig/ Directory](#)

(see also [network](#))

[/etc/sysconfig/networking/](#) [directory](#), [Directories in the /etc/sysconfig/ Directory](#)

[/etc/sysconfig/ntpd](#), [/etc/sysconfig/ntpd](#)

[/etc/sysconfig/quagga](#), [/etc/sysconfig/quagga](#)

[/etc/sysconfig/radvd](#), [/etc/sysconfig/radvd](#)

[/etc/sysconfig/rhn/](#) [directory](#), [Directories in the /etc/sysconfig/ Directory](#)

[/etc/sysconfig/samba](#), [/etc/sysconfig/samba](#)

[/etc/sysconfig/saslauthd](#), [/etc/sysconfig/saslauthd](#)

[/etc/sysconfig/selinux](#), [/etc/sysconfig/selinux](#)

[/etc/sysconfig/sendmail](#), [/etc/sysconfig/sendmail](#)

[/etc/sysconfig/spamassassin](#), [/etc/sysconfig/spamassassin](#)

[/etc/sysconfig/squid](#), [/etc/sysconfig/squid](#)

[/etc/sysconfig/system-config-users](#), [/etc/sysconfig/system-config-users](#)

[/etc/sysconfig/vncservers](#), [/etc/sysconfig/vncservers](#)

[/etc/sysconfig/xinetd](#), [/etc/sysconfig/xinetd](#)

additional information about, [The sysconfig Directory](#)

additional resources, [Additional Resources](#)

installed documentation, [Installed Documentation](#)

directories in, [Directories in the /etc/sysconfig/ Directory](#)

files found in, [Files in the /etc/sysconfig/ Directory](#)

## sysctl

configuring with [/etc/sysctl.conf](#), [Using the sysctl Command](#)

controlling [/proc/sys/](#), [Using the sysctl Command](#)

## SysRq (see system request key)

## 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 request key

enabling, [/proc/sys/](#)

System Request Key

definition of, [/proc/sys/](#)

setting timing for, [/proc/sys/kernel/](#)

system-config-authentication (see [Authentication Configuration Tool](#))

system-config-date (see [time configuration](#), [date configuration](#))

system-config-kdump (see [kdump](#))

system-config-services (see [services configuration](#))

systems

registration, [Registering the System and Managing Subscriptions](#)

subscription management, [Registering the System and Managing Subscriptions](#)

## T

testparm program, [Samba Distribution Programs](#)

time configuration

date, [Date and Time Setup](#)

synchronize with NTP server, [Network Time Protocol Properties](#), [Network Time Protocol Setup](#)

system-config-date, [Date and Time Properties](#)

time zone configuration, [Time Zone Properties](#)

TLB cache (see [hugepages](#))

TLS , [Setting Up an SSL Server](#)

(see also [Apache HTTP Server](#) )

tool

[Authentication Configuration Tool](#), [Configuring System Authentication](#)

top, [Using the top Command](#)

twm, [Window Managers](#)

(see also [X](#))

## U

updating currently installed packages

available updates, [Updating Packages with Software Update](#)

updating packages with PackageKit

PolicyKit, [Updating Packages with Software Update](#)

users



additional resources, [Additional Resources](#)  
installed documentation, [Installed Documentation](#)

## V

virtual file system (see [proc file system](#))

virtual files (see [proc file system](#))

virtual host (see [Apache HTTP Server](#))

### vsftpd

additional resources, [Additional Resources](#)  
installed documentation, [Installed Documentation](#)  
online documentation, [Online Documentation](#)

condrestart, [Starting and Stopping vsftpd](#)

#### configuration file

[/etc/vsftpd/vsftpd.conf](#), [vsftpd Configuration Options](#)

access controls, [Log In Options and Access Controls](#)

anonymous user options, [Anonymous User Options](#)

daemon options, [Daemon Options](#)

directory options, [Directory Options](#)

file transfer options, [File Transfer Options](#)

format of, [vsftpd Configuration Options](#)

local-user options, [Local-User Options](#)

logging options, [Logging Options](#)

login options, [Log In Options and Access Controls](#)

network options, [Network Options](#)

security options, [Security Options](#)

encrypting, [Encrypting vsftpd Connections Using TLS](#)

multihome configuration, [Starting Multiple Copies of vsftpd](#)

restarting, [Starting and Stopping vsftpd](#)

#### RPM

files installed by, [Files Installed with vsftpd](#)

securing, [Encrypting vsftpd Connections Using TLS](#), [SELinux Policy for vsftpd](#)

SELinux, [SELinux Policy for vsftpd](#)

starting, [Starting and Stopping vsftpd](#)

starting multiple copies of, [Starting Multiple Copies of vsftpd](#)

status, [Starting and Stopping vsftpd](#)

stopping, [Starting and Stopping vsftpd](#)

TLS, [Encrypting vsftpd Connections Using TLS](#)

## W

wbinfo program, [Samba Distribution Programs](#)

web server (see [Apache HTTP Server](#))

window managers (see [X](#))

**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](#)

display managers

configuration of preferred, [Runlevel 5](#)

definition of, [Runlevel 5](#)

GNOME, [Runlevel 5](#)

KDE, [Runlevel 5](#)

prefdm script, [Runlevel 5](#)

xdm, [Runlevel 5](#)

**fonts**

Fontconfig, [Fonts](#)

Fontconfig, adding fonts to, [Adding Fonts to Fontconfig](#)

FreeType, [Fonts](#)

introducing, [Fonts](#)

Xft, [Fonts](#)

introducing, [The X Window System](#)

**runlevels**

3, [Runlevel 3](#)

5, [Runlevel 5](#)

runlevels and, [Runlevels and X](#)

**window managers**

kwin, [Window Managers](#)

metacity, [Window Managers](#)

mwm, [Window Managers](#)

twm, [Window Managers](#)

X clients, [The X Window System, Desktop Environments and Window Managers](#)

desktop environments, [Desktop Environments](#)

startx command, [Runlevel 3](#)

window managers, [Window Managers](#)

xinit command, [Runlevel 3](#)

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)

xinit (see X)

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](#)

enabling plug-ins, [Enabling, Configuring, and Disabling Yum Plug-ins](#)

installing a package group with Yum, [Installing Packages](#)

installing with Yum, [Installing Packages](#)

listing packages with Yum

    Glob expressions, [Listing Packages](#)

    yum grouplist, [Listing Packages](#)

    yum list, [Listing Packages](#)

    yum list all, [Listing Packages](#)

    yum list available, [Listing Packages](#)

    yum list installed, [Listing Packages](#)

    yum repolist, [Listing Packages](#)

packages and package groups, [Packages and Package Groups](#)

plug-ins

    kabi, [Plug-in Descriptions](#)

    presto, [Plug-in Descriptions](#)

    product-id, [Plug-in Descriptions](#)

    refresh-packagekit, [Plug-in Descriptions](#)

    rhnplugin, [Plug-in Descriptions](#)

    search-disabled-repos, [Plug-in Descriptions](#)

    security, [Plug-in Descriptions](#)

    subscription-manager, [Plug-in Descriptions](#)

    yum-downloadonly, [Plug-in Descriptions](#)

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 package groups with Yum, [Removing Packages](#)

uninstalling packages with Yum, [Removing Packages](#)

    yum remove package\_name, [Removing Packages](#)

variables, [Using Yum Variables](#)

yum cache, [Working with Yum Cache](#)

yum clean, [Working with Yum Cache](#)

Yum plug-ins, [Yum Plug-ins](#)

Yum repositories

    configuring Yum and Yum repositories, [Configuring Yum and Yum Repositories](#)

yum update, [Upgrading the System Off-line with ISO and Yum](#)

Yum repositories

viewing Yum repositories with PackageKit, [Refreshing Software Sources \(Yum Repositories\)](#)

### 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 packages automatically, [Updating Packages](#)

updating security-related packages, [Updating Packages](#)