

Fedora 26

System

Administrator's Guide

Deployment, Configuration, and Administration of Fedora 26



Stephen Wadeley

Jaromír Hradílek

Petr Bokoč

Petr Kovář

Tomáš Čapek

Douglas Silas

Martin Prpič

Eliška Slobodová

Miroslav Svoboda

John Ha

David O'Brien

Michael Hideo

Don Domingo

Fedora 26 System Administrator's Guide

Deployment, Configuration, and Administration of Fedora 26

Edition 26

Author	Stephen Wadeley	swadeley@redhat.com
Author	Jaromír Hradílek	jhradilek@redhat.com
Author	Petr Bokoč	pbokoc@redhat.com
Author	Petr Kovář	pkovar@redhat.com
Author	Tomáš Čapek	tcapek@redhat.com
Author	Douglas Silas	silas@redhat.com
Author	Martin Prpič	
Author	Eliška Slobodová	
Author	Miroslav Svoboda	
Author	John Ha	
Author	David O'Brien	
Author	Michael Hideo	
Author	Don Domingo	

Copyright © 2017 Red Hat, Inc. and others.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at <http://creativecommons.org/licenses/by-sa/3.0/>. The original authors of this document, and Red Hat, designate the Fedora Project as the "Attribution Party" for purposes of CC-BY-SA. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

For guidelines on the permitted uses of the Fedora trademarks, refer to https://fedoraproject.org/wiki/Legal:Trademark_guidelines.

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

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

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

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

All other trademarks are the property of their respective owners.

The *System Administrator's Guide* documents relevant information regarding the deployment, configuration, and administration of Fedora 26. It is oriented towards system administrators with a basic understanding of the system.

Preface	xv
1. Target Audience	xv
2. How to Read this Book	xv
3. Document Conventions	xvii
3.1. Typographic Conventions	xvii
3.2. Pull-quote Conventions	xix
3.3. Notes and Warnings	xix
4. We Need Feedback!	xx
5. Acknowledgments	xx
 I. Basic System Configuration	 1
1. Opening Graphical Applications	3
1.1. Opening graphical applications from the command line	3
1.2. Launching Applications with Alt+F2	4
1.3. Launching applications from the Desktop Menu	7
1.3.1. Using GNOME menus	7
1.3.2. Using KDE menus	9
1.3.3. Using menus in LXDE, MATE, and XFCE	11
2. System Locale and Keyboard Configuration	15
2.1. Setting the System Locale	15
2.1.1. Displaying the Current Status	16
2.1.2. Listing Available Locales	16
2.1.3. Setting the Locale	16
2.2. Changing the Keyboard Layout	17
2.2.1. Displaying the Current Settings	17
2.2.2. Listing Available Keymaps	17
2.2.3. Setting the Keymap	17
2.3. Additional Resources	18
3. Configuring the Date and Time	19
3.1. Using the <code>timedatectl</code> Command	19
3.1.1. Displaying the Current Date and Time	19
3.1.2. Changing the Current Time	20
3.1.3. Changing the Current Date	20
3.1.4. Changing the Time Zone	20
3.1.5. Synchronizing the System Clock with a Remote Server	21
3.2. Using the <code>date</code> Command	21
3.2.1. Displaying the Current Date and Time	22
3.2.2. Changing the Current Time	23
3.2.3. Changing the Current Date	23
3.3. Using the <code>hwclock</code> Command	23
3.3.1. Displaying the Current Date and Time	24
3.3.2. Setting the Date and Time	24
3.3.3. Synchronizing the Date and Time	25
3.4. Additional Resources	26
4. Managing Users and Groups	27
4.1. Introduction to Users and Groups	27
4.1.1. User Private Groups	27
4.1.2. Shadow Passwords	27
4.2. Managing Users in a Graphical Environment	28
4.2.1. Using the Users Settings Tool	28
4.3. Using Command Line Tools	29

4.3.1. Adding a New User	30
4.3.2. Adding a New Group	33
4.3.3. Enabling Password Aging	33
4.3.4. Enabling Automatic Logouts	35
4.3.5. Creating Group Directories	36
4.4. Additional Resources	36
5. Gaining Privileges	39
5.1. The su Command	39
5.2. The sudo Command	40
5.3. Additional Resources	41
 II. Package Management	 43
6. DNF	45
6.1. Checking For and Updating Packages	45
6.1.1. Checking For Updates	45
6.1.2. Updating Packages	46
6.1.3. Preserving Configuration File Changes	47
6.2. Packages and Package Groups	48
6.2.1. Searching Packages	48
6.2.2. Listing Packages	48
6.2.3. Displaying Package Information	51
6.2.4. Installing Packages	52
6.2.5. Removing Packages	54
6.2.6. Working with Transaction History	55
6.3. Configuring DNF and DNF Repositories	57
6.3.1. Setting [main] Options	57
6.3.2. Setting [repository] Options	58
6.3.3. Using DNF Variables	59
6.4. Viewing the Current Configuration	60
6.5. Adding, Enabling, and Disabling a DNF Repository	60
6.6. Additional Resources	61
 III. Infrastructure Services	 63
7. Services and Daemons	65
7.1. Configuring Services	65
7.1.1. Enabling the Service	65
7.1.2. Disabling the Service	66
7.2. Running Services	66
7.2.1. Checking the Service Status	66
7.2.2. Running the Service	68
7.2.3. Stopping the Service	68
7.2.4. Restarting the Service	68
7.3. Additional Resources	69
7.3.1. Installed Documentation	69
7.3.2. Related Books	69
8. OpenSSH	71
8.1. The SSH Protocol	71
8.1.1. Why Use SSH?	71
8.1.2. Main Features	72
8.1.3. Protocol Versions	72
8.1.4. Event Sequence of an SSH Connection	73

8.2. Configuring OpenSSH	74
8.2.1. Configuration Files	75
8.2.2. Starting an OpenSSH Server	76
8.2.3. Requiring SSH for Remote Connections	77
8.2.4. Using Key-based Authentication	77
8.3. Using OpenSSH Certificate Authentication	80
8.3.1. Introduction to SSH Certificates	80
8.3.2. Support for SSH Certificates	81
8.3.3. Creating SSH CA Certificate Signing Keys	81
8.3.4. Distributing and Trusting SSH CA Public Keys	83
8.3.5. Creating SSH Certificates	84
8.3.6. Signing an SSH Certificate Using a PKCS#11 Token	89
8.3.7. Viewing an SSH CA Certificate	89
8.3.8. Revoking an SSH CA Certificate	90
8.4. OpenSSH Clients	91
8.4.1. Using the ssh Utility	91
8.4.2. Using the scp Utility	92
8.4.3. Using the sftp Utility	93
8.5. More Than a Secure Shell	94
8.5.1. X11 Forwarding	94
8.5.2. Port Forwarding	94
8.6. Additional Resources	95
9. TigerVNC	97
9.1. VNC Server	97
9.1.1. Installing VNC Server	97
9.1.2. Configuring VNC Server	97
9.1.3. Starting VNC Server	98
9.1.4. Terminating a VNC Session	99
9.2. VNC Viewer	99
9.2.1. Installing VNC Viewer	99
9.2.2. Connecting to VNC Server	100
9.2.3. Connecting to VNC Server Using SSH	101
9.3. Additional Resources	102
IV. Servers	103
10. Web Servers	105
10.1. The Apache HTTP Server	105
10.1.1. Notable Changes	105
10.1.2. Updating the Configuration	107
10.1.3. Running the httpd Service	108
10.1.4. Editing the Configuration Files	109
10.1.5. Working with Modules	140
10.1.6. Setting Up Virtual Hosts	141
10.1.7. Setting Up an SSL Server	141
10.1.8. Additional Resources	149
11. Mail Servers	151
11.1. Email Protocols	151
11.1.1. Mail Transport Protocols	151
11.1.2. Mail Access Protocols	151
11.2. Email Program Classifications	154
11.2.1. Mail Transport Agent	154
11.2.2. Mail Delivery Agent	154

11.2.3. Mail User Agent	155
11.3. Mail Transport Agents	155
11.3.1. Postfix	155
11.3.2. Sendmail	157
11.3.3. Fetchmail	162
11.3.4. Mail Transport Agent (MTA) Configuration	166
11.4. Mail Delivery Agents	167
11.4.1. Procmal Configuration	167
11.4.2. Procmal Recipes	168
11.5. Mail User Agents	173
11.5.1. Securing Communication	173
11.6. Additional Resources	175
11.6.1. Installed Documentation	175
11.6.2. Useful Websites	176
11.6.3. Related Books	176
12. Directory Servers	179
12.1. OpenLDAP	179
12.1.1. Introduction to LDAP	179
12.1.2. Installing the OpenLDAP Suite	181
12.1.3. Configuring an OpenLDAP Server	183
12.1.4. SELinux Policy for Applications Using LDAP	193
12.1.5. Running an OpenLDAP Server	193
12.1.6. Configuring a System to Authenticate Using OpenLDAP	194
12.1.7. Additional Resources	195
12.1.8. Related Books	196
13. File and Print Servers	197
13.1. Samba	197
13.1.1. Introduction to Samba	197
13.1.2. Samba Daemons and Related Services	198
13.1.3. Connecting to a Samba Share	199
13.1.4. Mounting the Share	200
13.1.5. Configuring a Samba Server	201
13.1.6. Starting and Stopping Samba	202
13.1.7. Samba Server Types and the smb.conf File	203
13.1.8. Samba Security Modes	211
13.1.9. Samba Account Information Databases	213
13.1.10. Samba Network Browsing	214
13.1.11. Samba with CUPS Printing Support	215
13.1.12. Samba Distribution Programs	216
13.1.13. Additional Resources	220
13.2. FTP	221
13.2.1. The File Transfer Protocol	221
13.2.2. FTP Servers	222
13.2.3. Files Installed with vsftpd	223
13.2.4. Starting and Stopping vsftpd	223
13.2.5. vsftpd Configuration Options	224
13.2.6. Additional Resources	233
13.3. Printer Configuration	234
13.3.1. Starting the Printers Configuration Tool	234
13.3.2. Starting Printer Setup	235
13.3.3. Adding a Local Printer	235
13.3.4. Adding an AppSocket/HP JetDirect printer	236
13.3.5. Adding an IPP Printer	237

13.3.6. Adding an LPD/LPR Host or Printer	238
13.3.7. Adding a Samba (SMB) printer	239
13.3.8. Selecting the Printer Model and Finishing	241
13.3.9. Printing a Test Page	244
13.3.10. Modifying Existing Printers	245
13.3.11. Additional Resources	251
14. Configuring NTP Using the chrony Suite	253
14.1. Introduction to the chrony Suite	253
14.1.1. Differences Between ntpd and chronyd	253
14.1.2. Choosing Between NTP Daemons	254
14.2. Understanding chrony and Its Configuration	254
14.2.1. Understanding chronyd	254
14.2.2. Understanding chronyc	254
14.2.3. Understanding the chrony Configuration Commands	255
14.2.4. Security with chronyc	258
14.3. Using chrony	259
14.3.1. Installing chrony	259
14.3.2. Checking the Status of chronyd	260
14.3.3. Starting chronyd	260
14.3.4. Stopping chronyd	260
14.3.5. Checking if chrony is Synchronized	260
14.3.6. Manually Adjusting the System Clock	264
14.4. Setting Up chrony for Different Environments	264
14.4.1. Setting Up chrony for a System Which is Infrequently Connected	264
14.4.2. Setting Up chrony for a System in an Isolated Network	265
14.5. Using chronyc	265
14.5.1. Using chronyc to Control chronyd	265
14.5.2. Using chronyc for Remote Administration	266
14.6. Additional Resources	267
14.6.1. Installed Documentation	267
14.6.2. Online Documentation	267
15. Configuring NTP Using ntpd	269
15.1. Introduction to NTP	269
15.2. NTP Strata	269
15.3. Understanding NTP	270
15.4. Understanding the Drift File	271
15.5. UTC, Timezones, and DST	271
15.6. Authentication Options for NTP	272
15.7. Managing the Time on Virtual Machines	272
15.8. Understanding Leap Seconds	272
15.9. Understanding the ntpd Configuration File	273
15.10. Understanding the ntpd Sysconfig File	274
15.11. Disabling chrony	275
15.12. Checking if the NTP Daemon is Installed	275
15.13. Installing the NTP Daemon (ntpd)	275
15.14. Checking the Status of NTP	275
15.15. Configure the Firewall to Allow Incoming NTP Packets	276
15.15.1. Change the Firewall Settings	276
15.15.2. Open Ports in the Firewall for NTP Packets	276
15.16. Configure ntpdate Servers	276
15.17. Configure NTP	277
15.17.1. Configure Access Control to an NTP Service	277
15.17.2. Configure Rate Limiting Access to an NTP Service	278

15.17.3. Adding a Peer Address	279
15.17.4. Adding a Server Address	279
15.17.5. Adding a Broadcast or Multicast Server Address	279
15.17.6. Adding a Multicast Client Address	280
15.17.7. Adding a Broadcast Client Address	280
15.17.8. Adding a Multicast Server Address	280
15.17.9. Adding a Multicast Client Address	280
15.17.10. Configuring the Burst Option	281
15.17.11. Configuring the iburst Option	281
15.17.12. Configuring Symmetric Authentication Using a Key	281
15.17.13. Configuring the Poll Interval	281
15.17.14. Configuring Server Preference	282
15.17.15. Configuring the Time-to-Live for NTP Packets	282
15.17.16. Configuring the NTP Version to Use	282
15.18. Configuring the Hardware Clock Update	282
15.19. Configuring Clock Sources	283
15.20. Additional Resources	283
15.20.1. Installed Documentation	283
15.20.2. Useful Websites	284
16. Configuring PTP Using ptp4l	285
16.1. Introduction to PTP	285
16.1.1. Understanding PTP	285
16.1.2. Advantages of PTP	287
16.2. Using PTP	287
16.2.1. Checking for Driver and Hardware Support	287
16.2.2. Installing PTP	288
16.2.3. Starting ptp4l	288
16.3. Specifying a Configuration File	290
16.4. Using the PTP Management Client	290
16.5. Synchronizing the Clocks	291
16.6. Verifying Time Synchronization	292
16.7. Serving PTP Time with NTP	294
16.8. Serving NTP Time with PTP	294
16.9. Synchronize to PTP or NTP Time Using timemaster	295
16.9.1. Starting timemaster as a Service	295
16.9.2. Understanding the timemaster Configuration File	295
16.9.3. Configuring timemaster Options	297
16.10. Improving Accuracy	298
16.11. Additional Resources	298
16.11.1. Installed Documentation	298
16.11.2. Useful Websites	299
V. Monitoring and Automation	301
17. System Monitoring Tools	303
17.1. Viewing System Processes	303
17.1.1. Using the ps Command	303
17.1.2. Using the top Command	304
17.1.3. Using the System Monitor Tool	305
17.2. Viewing Memory Usage	306
17.2.1. Using the free Command	306
17.2.2. Using the System Monitor Tool	307
17.3. Viewing CPU Usage	308

17.3.1. Using the System Monitor Tool	308
17.4. Viewing Block Devices and File Systems	309
17.4.1. Using the lsblk Command	309
17.4.2. Using the blkid Command	310
17.4.3. Using the partx Command	311
17.4.4. Using the findmnt Command	311
17.4.5. Using the df Command	312
17.4.6. Using the du Command	313
17.4.7. Using the System Monitor Tool	314
17.5. Viewing Hardware Information	315
17.5.1. Using the lspci Command	315
17.5.2. Using the lsusb Command	315
17.5.3. Using the lspcmcia Command	316
17.5.4. Using the lscpu Command	317
17.6. Monitoring Performance with Net-SNMP	317
17.6.1. Installing Net-SNMP	318
17.6.2. Running the Net-SNMP Daemon	318
17.6.3. Configuring Net-SNMP	319
17.6.4. Retrieving Performance Data over SNMP	322
17.6.5. Extending Net-SNMP	325
17.7. Additional Resources	330
17.7.1. Installed Documentation	330
18. Viewing and Managing Log Files	331
18.1. Locating Log Files	331
18.2. Basic Configuration of Rsyslog	331
18.2.1. Filters	332
18.2.2. Actions	335
18.2.3. Templates	340
18.2.4. Global Directives	343
18.2.5. Log Rotation	344
18.3. Using the New Configuration Format	345
18.3.1. Rulesets	346
18.3.2. Compatibility with syslogd	347
18.4. Working with Queues in Rsyslog	347
18.4.1. Defining Queues	349
18.4.2. Managing Queues	352
18.5. Configuring rsyslog on a Logging Server	354
18.5.1. Using The New Template Syntax on a Logging Server	356
18.6. Using Rsyslog Modules	356
18.6.1. Importing Text Files	358
18.6.2. Exporting Messages to a Database	359
18.6.3. Enabling Encrypted Transport	359
18.6.4. Using RELP	359
18.7. Interaction of Rsyslog and Journal	359
18.8. Structured Logging with Rsyslog	360
18.8.1. Importing Data from Journal	361
18.8.2. Filtering Structured Messages	362
18.8.3. Parsing JSON	362
18.8.4. Storing Messages in the MongoDB	363
18.9. Debugging Rsyslog	363
18.10. Troubleshooting Logging to a Server	364
18.11. Using the Journal	364
18.11.1. Viewing Log Files	364

18.11.2. Access Control	366
18.11.3. Using The Live View	366
18.11.4. Filtering Messages	366
18.11.5. Enabling Persistent Storage	369
18.12. Managing Log Files in a Graphical Environment	369
18.12.1. Viewing Log Files	369
18.12.2. Adding a Log File	372
18.12.3. Monitoring Log Files	373
18.13. Additional Resources	374
19. Automating System Tasks	377
19.1. Cron and Anacron	377
19.1.1. Installing Cron and Anacron	377
19.1.2. Running the Crond Service	378
19.1.3. Configuring Anacron Jobs	378
19.1.4. Configuring Cron Jobs	380
19.1.5. Controlling Access to Cron	382
19.1.6. Black and White Listing of Cron Jobs	382
19.2. At and Batch	382
19.2.1. Installing At and Batch	382
19.2.2. Running the At Service	383
19.2.3. Configuring an At Job	384
19.2.4. Configuring a Batch Job	385
19.2.5. Viewing Pending Jobs	385
19.2.6. Additional Command Line Options	385
19.2.7. Controlling Access to At and Batch	385
19.3. Additional Resources	386
20. OProfile	387
20.1. Overview of Tools	387
20.1.1. operf vs. opcontrol	388
20.2. Using operf	389
20.2.1. Specifying the Kernel	389
20.2.2. Setting Events to Monitor	389
20.2.3. Categorization of Samples	391
20.3. Configuring OProfile Using Legacy Mode	391
20.3.1. Specifying the Kernel	391
20.3.2. Setting Events to Monitor	392
20.3.3. Separating Kernel and User-space Profiles	396
20.4. Starting and Stopping OProfile Using Legacy Mode	397
20.5. Saving Data in Legacy Mode	397
20.6. Analyzing the Data	398
20.6.1. Using opreport	399
20.6.2. Using opreport on a Single Executable	400
20.6.3. Getting More Detailed Output on the Modules	401
20.6.4. Using opannotate	402
20.7. Understanding the /dev/oprofile/ directory	403
20.8. Example Usage	403
20.9. OProfile Support for Java	403
20.9.1. Profiling Java Code	404
20.10. Graphical Interface	404
20.11. OProfile and SystemTap	406
20.12. Additional Resources	407

VI. Kernel, Module and Driver Configuration	409
21. Working with the GRUB 2 Boot Loader	411
21.1. Introduction to GRUB 2	411
21.2. Configuring the GRUB 2 Boot Loader	412
21.3. Making Temporary Changes to a GRUB 2 Menu	412
21.4. Making Persistent Changes to a GRUB 2 Menu Using the grubby Tool	413
21.5. Customizing the GRUB 2 Configuration File	415
21.5.1. Changing the Default Boot Entry	415
21.5.2. Editing a Menu Entry	416
21.5.3. Adding a new Entry	417
21.5.4. Creating a Custom Menu	417
21.6. GRUB 2 Password Protection	419
21.6.1. Setting Up Users and Password Protection, Specifying Menu Entries	419
21.6.2. Password Encryption	421
21.7. Reinstalling GRUB 2	421
21.7.1. Reinstalling GRUB 2 on BIOS-Based Machines	421
21.7.2. Reinstalling GRUB 2 on UEFI-Based Machines	422
21.7.3. Resetting and Reinstalling GRUB 2	422
21.8. GRUB 2 over a Serial Console	422
21.8.1. Configuring the GRUB 2 Menu	422
21.8.2. Using screen to Connect to the Serial Console	423
21.9. Terminal Menu Editing During Boot	424
21.9.1. Booting to Rescue Mode	424
21.9.2. Booting to Emergency Mode	424
21.9.3. Changing and Resetting the Root Password	425
21.10. UEFI Secure Boot	428
21.10.1. UEFI Secure Boot Support in Fedora	428
21.11. Additional Resources	428
22. Manually Upgrading the Kernel	431
22.1. Overview of Kernel Packages	431
22.2. Preparing to Upgrade	432
22.3. Downloading the Upgraded Kernel	433
22.4. Performing the Upgrade	433
22.5. Verifying the Initial RAM Disk Image	434
22.6. Verifying the Boot Loader	436
22.6.1. Configuring the GRUB 2 Boot Loader	437
22.6.2. Configuring the OS/400 Boot Loader	438
22.6.3. Configuring the YABOOT Boot Loader	438
23. Working with Kernel Modules	441
23.1. Listing Currently-Loaded Modules	441
23.2. Displaying Information About a Module	442
23.3. Loading a Module	444
23.4. Unloading a Module	445
23.5. Setting Module Parameters	446
23.6. Persistent Module Loading	447
23.7. Signing Kernel Modules for Secure Boot	448
23.7.1. Prerequisites	448
23.7.2. Kernel Module Authentication	449
23.7.3. Generating a Public and Private X.509 Key Pair	450
23.7.4. Enrolling Public Key on Target System	451
23.7.5. Signing Kernel Module with the Private Key	452
23.7.6. Loading Signed Kernel Module	453

23.8. Additional Resources	453
A. RPM	455
A.1. RPM Design Goals	455
A.2. Using RPM	456
A.2.1. Installing and Upgrading Packages	456
A.2.2. Uninstalling Packages	459
A.2.3. Freshening Packages	460
A.2.4. Querying Packages	461
A.2.5. Verifying Packages	461
A.3. Finding and Verifying RPM Packages	462
A.3.1. Finding RPM Packages	462
A.3.2. Checking Package Signatures	463
A.4. Common Examples of RPM Usage	464
A.5. Additional Resources	464
B. Revision History	467
Index	469

Preface

The *System Administrator's Guide* contains information on how to customize the Fedora 26 system to fit your needs. If you are looking for a comprehensive, task-oriented guide for configuring and customizing your system, this is the manual for you.

This manual discusses many intermediate topics such as the following:

- Installing and managing packages using **DNF**
- Configuring **Apache HTTP Server**, **Postfix**, **Sendmail** and other enterprise-class servers and software
- Working with kernel modules and upgrading the kernel



Note

Some of the graphical procedures and menu locations are specific to GNOME, but most command line instructions will be universally applicable.

1. Target Audience

The *System Administrator's Guide* assumes you have a basic understanding of the Fedora operating system. If you need help with the installation of this system, refer to the [Fedora Installation Guide](#)¹.

2. How to Read this Book

This manual is divided into the following main categories:

Part I, "Basic System Configuration"

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

Chapter 1, Opening Graphical Applications describes methods for opening **Graphical User Interface**, or *GUI*, applications in various environments.

Chapter 2, System Locale and Keyboard Configuration covers basic language and keyboard setup. Read this chapter if you need to configure the language of your desktop, change the keyboard layout, or add the keyboard layout indicator to the panel.

Chapter 3, Configuring the Date and Time covers the configuration of the system date and time. Read this chapter if you need to set or change the date and time.

Chapter 4, Managing Users and Groups covers the management of users and groups in a graphical user interface and on the command line. Read this chapter if you need to manage users and groups on your system, or enable password aging.

Chapter 5, Gaining Privileges covers ways to gain administrative privileges using *setuid* programs such as **su** and **sudo**.

¹ <http://docs.fedoraproject.org/install-guide>

Part II, “Package Management”

This part describes how to manage software packages on Fedora using **DNF**.

Chapter 6, DNF describes the **DNF** package manager. Read this chapter for information how to search, install, update, and uninstall packages on the command line.

Part III, “Infrastructure Services”

This part provides information on how to configure services and daemons, configure authentication, and enable remote logins.

Chapter 7, Services and Daemons covers the configuration of the services to be run when a system is started, and provides information on how to start, stop, and restart the services on the command line using the **systemctl** utility.

Chapter 8, OpenSSH describes how to enable a remote login via the SSH protocol. It covers the configuration of the **sshd** service, as well as a basic usage of the **ssh**, **scp**, **sftp** client utilities. Read this chapter if you need a remote access to a machine.

Chapter 9, TigerVNC describes the *virtual network computing* (VNC) method of graphical desktop sharing which allows you to remotely control other computers.

Part IV, “Servers”

This part discusses various topics related to servers such as how to set up a Web server or share files and directories over the network.

Chapter 10, Web Servers focuses on the **Apache HTTP Server 2.2**, a robust, full-featured open source web server developed by the Apache Software Foundation. Read this chapter if you need to configure a web server on your system.

Chapter 11, Mail Servers reviews modern email protocols in use today, and some of the programs designed to send and receive email, including **Postfix**, **Sendmail**, **Fetchmail**, and **Procmail**. Read this chapter if you need to configure a mail server on your system.

Chapter 12, Directory Servers covers the installation and configuration of **OpenLDAP 2.4**, an open source implementation of the LDAPv2 and LDAPv3 protocols. Read this chapter if you need to configure a directory server on your system.

Chapter 13, File and Print Servers guides you through the installation and configuration of **Samba**, an open source implementation of the Server Message Block (SMB) protocol, and **vsftpd**, the primary FTP server shipped with Fedora. Additionally, it explains how to use the **Printer Configuration** tool to configure printers. Read this chapter if you need to configure a file or print server on your system.

Chapter 14, Configuring NTP Using the chrony Suite covers the installation and configuration of the **chrony** suite, a client and a server for the Network Time Protocol (NTP). Read this chapter if you need to configure the system to synchronize the clock with a remote NTP server, or set up an NTP server on this system.

Chapter 15, Configuring NTP Using ntpd covers the installation and configuration of the NTP daemon, **ntpd**, for the Network Time Protocol (NTP). Read this chapter if you need to configure the system to synchronize the clock with a remote NTP server, or set up an NTP server on this system, and you prefer not to use the **chrony** application.

Chapter 16, Configuring PTP Using ptp4i covers the installation and configuration of the Precision Time Protocol application, **ptp4i**, an application for use with network drivers that support the Precision Network Time Protocol (PTP). Read this chapter if you need to configure the system to synchronize the system clock with a master PTP clock.

Part V, “Monitoring and Automation”

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

Chapter 17, System Monitoring Tools discusses applications and commands that can be used to retrieve important information about the system. Read this chapter to learn how to gather essential system information.

Chapter 18, Viewing and Managing Log Files describes the configuration of the `rsyslog` daemon, and explains how to locate, view, and monitor log files. Read this chapter to learn how to work with log files.

Chapter 19, Automating System Tasks provides an overview of the **cron**, **at**, and **batch** utilities. Read this chapter to learn how to use these utilities to perform automated tasks.

Chapter 20, OProfile covers **OProfile**, a low overhead, system-wide performance monitoring tool. Read this chapter for information on how to use **OProfile** on your system.

Part VI, “Kernel, Module and Driver Configuration”

This part covers various tools that assist administrators with kernel customization.

Chapter 21, Working with the GRUB 2 Boot Loader describes the GNU GRand Unified Boot loader (GRUB) version 2 boot loader, which enables selecting an operating system or kernel to be loaded at system boot time.

Chapter 22, Manually Upgrading the Kernel provides important information on how to manually update a kernel package using the **rpm** command instead of **dnf**. Read this chapter if you cannot update a kernel package with the **DNF** package manager.

Chapter 23, Working with Kernel Modules explains how to display, query, load, and unload kernel modules and their dependencies, and how to set module parameters. Additionally, it covers specific kernel module capabilities such as using multiple Ethernet cards and using channel bonding. Read this chapter if you need to work with kernel modules.

Appendix A, RPM

This appendix concentrates on the RPM Package Manager (RPM), an open packaging system used by Fedora, and the use of the **rpm** utility. Read this appendix if you need to use **rpm** instead of **dnf**.

3. Document Conventions

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

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts*² set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

3.1. Typographic Conventions

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

² <https://fedorahosted.org/liberation-fonts/>

Mono-spaced Bold

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

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

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

Key combinations can be distinguished from keycaps by the hyphen connecting each part of a key combination. For example:

Press **Enter** to execute the command.

Press **Ctrl+Alt+F2** to switch to the first virtual terminal. Press **Ctrl+Alt+F1** to return to your X-Windows session.

The first paragraph highlights the particular keycap to press. The second highlights two key combinations (each a set of three keycaps with each set pressed simultaneously).

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

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

Proportional Bold

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

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

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

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

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

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

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

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

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

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

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

Publican is a *DocBook* publishing system.

3.2. Pull-quote Conventions

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

Output sent to a terminal is set in **mono-spaced roman** and presented thus:

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

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

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
    public static void main(String args[])
        throws Exception
    {
        InitialContext iniCtx = new InitialContext();
        Object          ref    = iniCtx.lookup("EchoBean");
        EchoHome        home   = (EchoHome) ref;
        Echo             echo   = home.create();

        System.out.println("Created Echo");

        System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
    }
}
```

3.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.



Note

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



Important

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



Warning

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

4. We Need Feedback!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please open an issue on Pagure: <https://pagure.io/system-administrators-guide/issues>

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

5. Acknowledgments

Certain portions of this text first appeared in the *Red Hat Enterprise Linux 7 System Administrator's Guide*, copyright © 2014–2017 Red Hat, Inc., available at https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/System_Administrators_Guide/index.html.

Section 17.6, “Monitoring Performance with Net-SNMP” is based on an article written by Michael Solberg.

The authors of this book would like to thank the following people for their valuable contributions: Adam Tkáč, Andrew Fitzsimon, Andrius Benokraitis, Brian Cleary Edward Bailey, Garrett LeSage, Jeffrey Fearn, Joe Orton, Joshua Wulf, Karsten Wade, Lucy Ringland, Marcela Mašláňová, Mark Johnson, Michael Behm, Miroslav Lichvár, Radek Vokál, Rahul Kavalapara, Rahul Sundaram, Sandra Moore, Zbyšek Mráz, Jan Včelák, Peter Hutterer, T.C. Hollingsworth, and James Antill, among many others.

Part I. Basic System Configuration

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

Opening Graphical Applications

Fedora provides graphical applications in addition to command line utilities for configuring many features. This chapter describes methods for opening **Graphical User Interface**, or **GUI**, applications in various environments.

1.1. Opening graphical applications from the command line

Graphical applications can be launched from a terminal window or console session by simply typing the name of the application.

```
[fedorauser@localhost]$ firefox
```



File names vs Application names

Programs are opened from the command line using the name of the executable file provided in the program's package. An entry in the desktop menu will often be named differently from the file it executes. For example, the GNOME disk management utility appears in the menu as **Disks**, and the file it executes is `/usr/bin/gnome-disks`.

When a program is executed on the command line, the terminal is occupied until the program completes. When a graphical application is executed from the command line, the program's error output, or **STDERR**, is sent to the terminal window. This can be especially useful when troubleshooting.

Example 1.1. Viewing errors by launching graphical applications from the command line

```
[fedorauser@localhost]$ astromenace-wrapper
AstroMenace 1.3.1 121212

Open XML file: /home/fedorauser/.config/astromenace/amconfig.xml
VFS file was opened /usr/share/astromenace/gamedata.vfs

Vendor      : OpenAL Community
Renderer    : OpenAL Soft
Version     : 1.1 ALSOFT 1.15.1
ALut ver    : 1.1

Font initialized: DATA/FONT/LiberationMono-Bold.ttf

Current Video Mode: 3200x1080 32bit

Xinerama/TwinView detected.
Screen count: 2
Screen #0: (0, 0) x (1920, 1080)
Screen #1: (1920, 0) x (1280, 1024)

Supported resolutions list:
640x480 16bit
640x480 32bit
640x480 0bit
768x480 16bit
<output truncated>
```


Chapter 1. Opening Graphical Applications

To launch a graphical application, but fork the additional output into the background and return the terminal for immediate use, use the shell's `job control` feature.

```
[fedorauser@localhost]$ emacs foo.txt &
```



Ending a session

Applications that hold the command line prompt until they complete will close when the terminal session ends, even if they are forked into the background.

GUI programs can also be launched on one TTY and displayed on another by specifying the `DISPLAY` variable. This can be useful when running multiple graphical sessions, or for troubleshooting problems with a desktop session.

1. Switch to another TTY using the key combination **Ctrl-Alt-F2** and log in. Note that consoles are available by default with **F2** through **F6**.
2. Identify the X session you want to target. The `DISPLAY` variable is always an integer preceded by a colon, and will be `:0` in most cases. Check the arguments of the currently running **X** process to verify the value. The command below shows both the `DISPLAY` variable as well as the TTY that **X** is running on, **tty1**.

```
[fedorauser@localhost]$ ps aux|grep /usr/bin/X
root      1498  7.1  1.0 521396 353984 tty1    Ss+  00:04  66:34 /usr/bin/X :0 vt1 -
background none -nolisten tcp -auth /var/run/kdm/A:0-22Degc

root      23874  0.0  0.0 109184   900 pts/21   S+   15:35   0:00 grep --color=auto /usr/
bin/X
```

3. Specify the `DISPLAY` variable when executing the program.

```
[fedorauser@localhost]$ DISPLAY=:0 gnome-shell --replace &
```

4. Switch back to the TTY the graphical session is running on. Since the example above shows **X** running on **vt1**, pressing **Ctrl+Alt+F1** will return to the desktop environment.

1.2. Launching Applications with Alt+F2

Most desktop environments follow the convention of using the key combination **Alt+F2** for opening new applications. Pressing **Alt+F2** brings up a prompt for a command to be entered into.

Commands entered into this dialog box function much as they would if entered in a terminal. Applications are known by their file name, and can accept arguments.

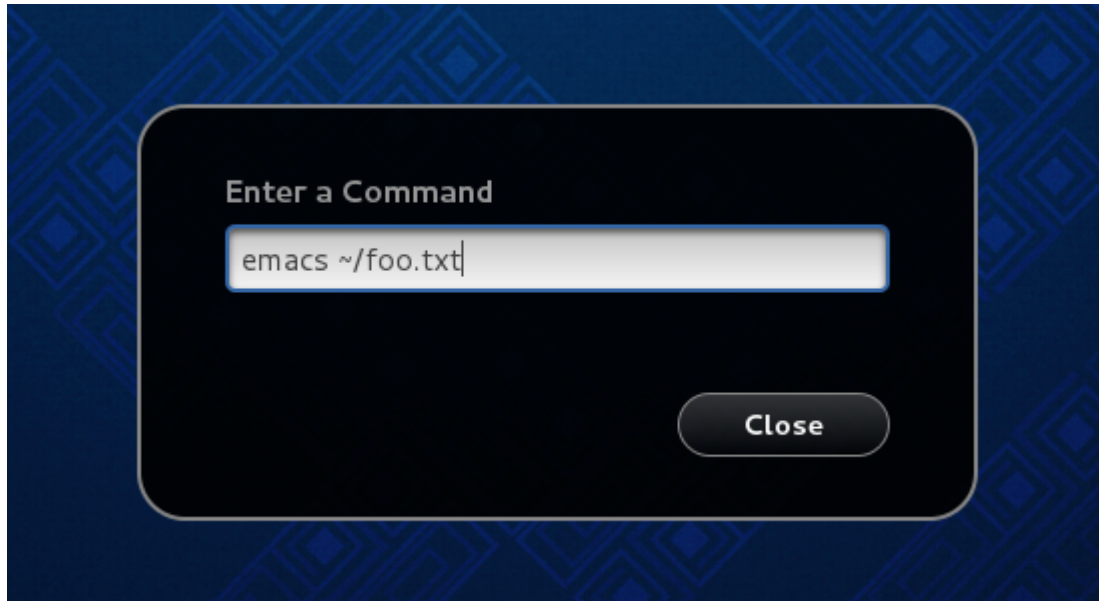


Figure 1.1. Using **Alt+F2** with **GNOME**

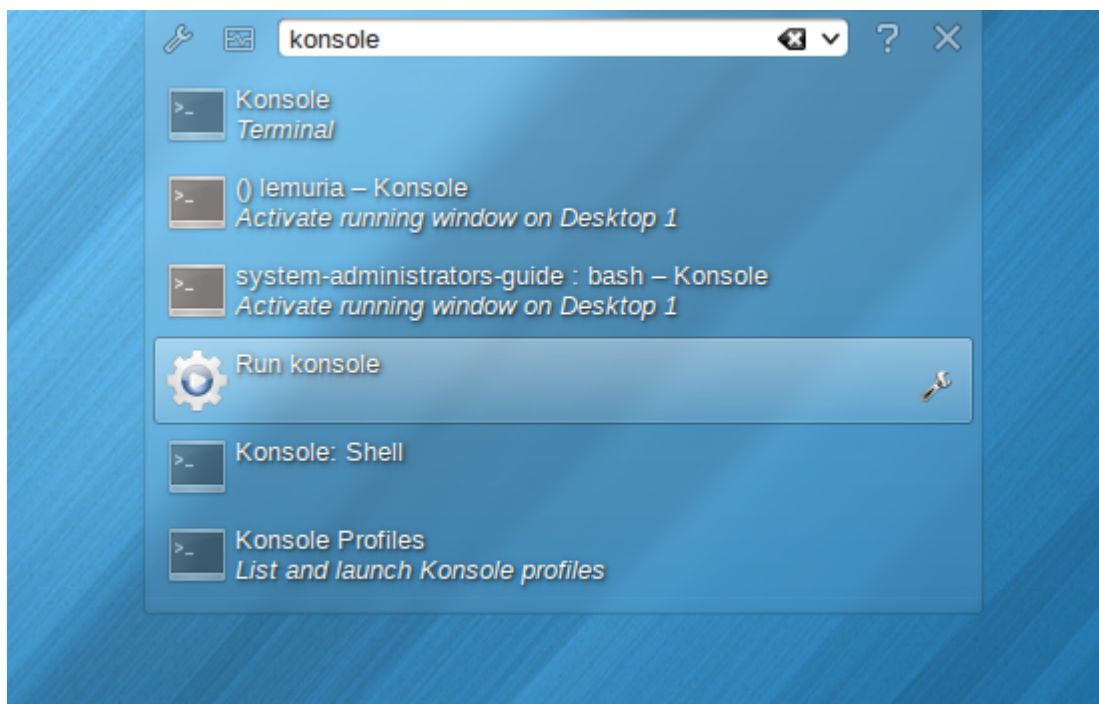


Figure 1.2. Using **Alt+F2** with **KDE**

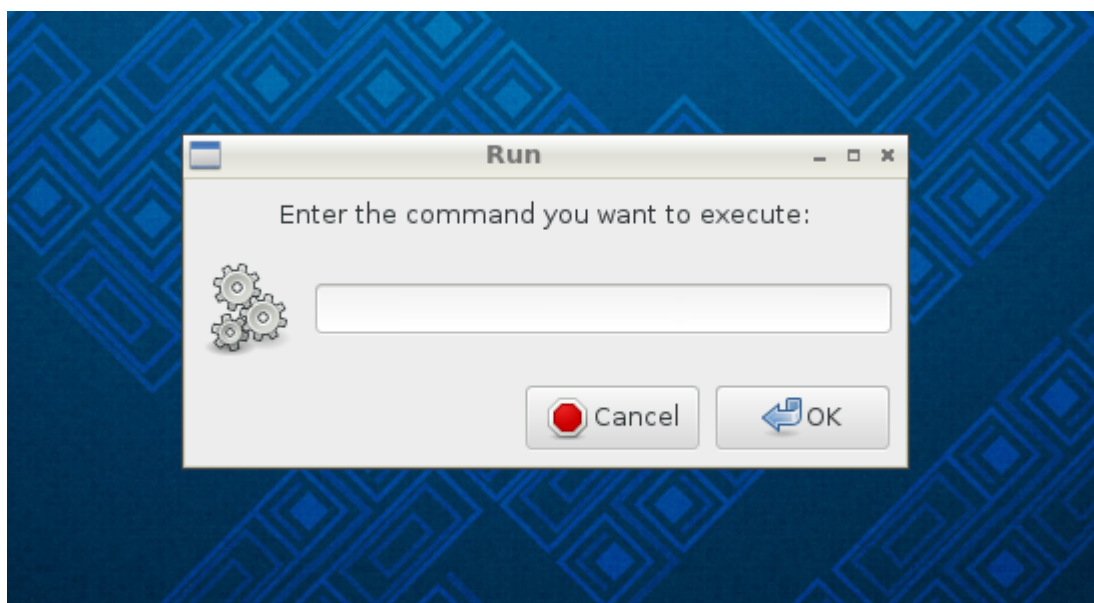


Figure 1.3. Using **Alt+F2** with **LXDE**

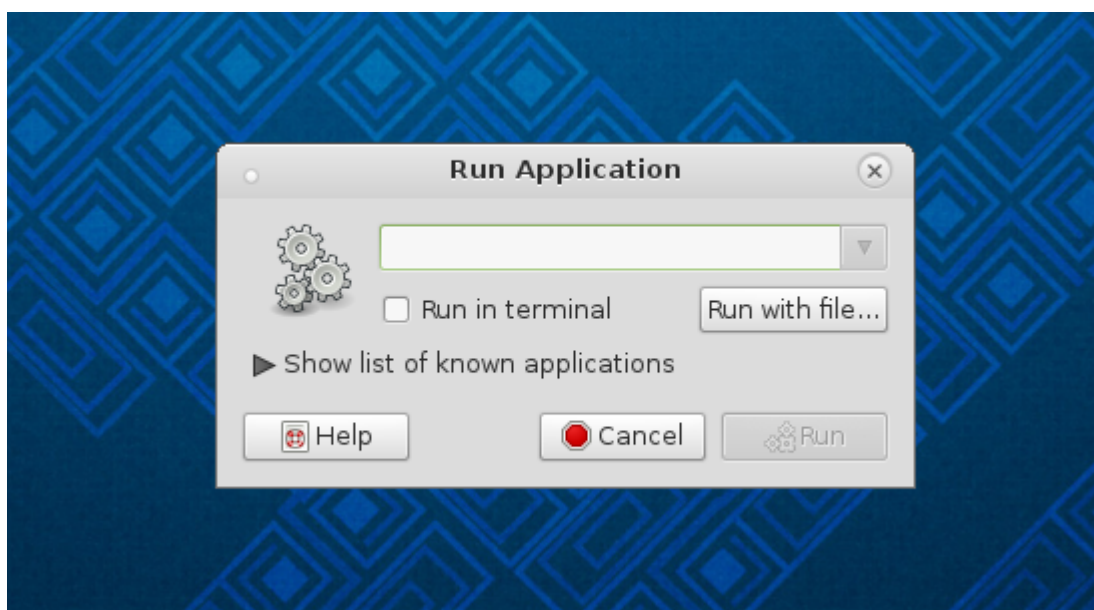


Figure 1.4. Using **Alt+F2** with **MATE**

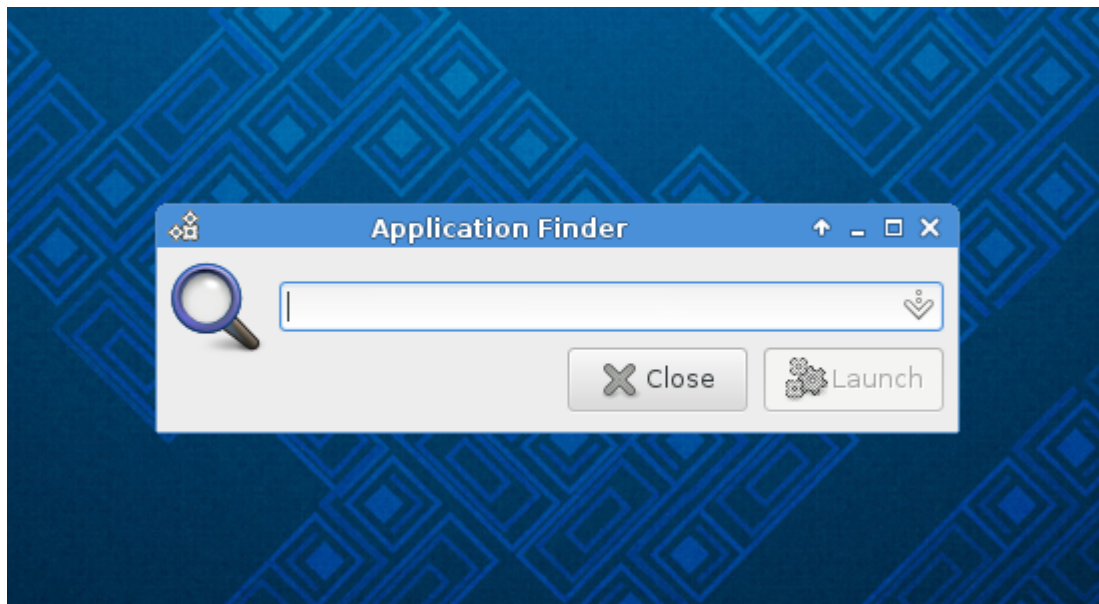


Figure 1.5. Using **Alt+F2** with XFCE

1.3. Launching applications from the Desktop Menu

Applications can also be opened from the menu system provided by the desktop environment in use. While the presentation may vary between desktop environments, the menu entries and their categories are provided by the individual application and standardized by the [freedesktop.org Desktop Menu Specification](http://standards.freedesktop.org/menu-spec/latest.html)¹. Some desktop environments also provide search functionality in their menu system to allow quick and easy access to applications.

1.3.1. Using GNOME menus

The GNOME menu, called the **overview**, can be accessed by either clicking the **Activities** button in the top left of the primary display, by moving the mouse past the top left hot corner, or by pressing the **Super (Windows)** key. The **overview** presents documents in addition to applications.

Selecting an item from the menu is best accomplished using the **search box**. Simply bring up the **overview**, and begin typing the name of the application you want to launch. Pressing enter will launch the highlighted application, or you can use the arrow keys or mouse to choose an alternative.

¹ <http://standards.freedesktop.org/menu-spec/menu-spec-latest.html>

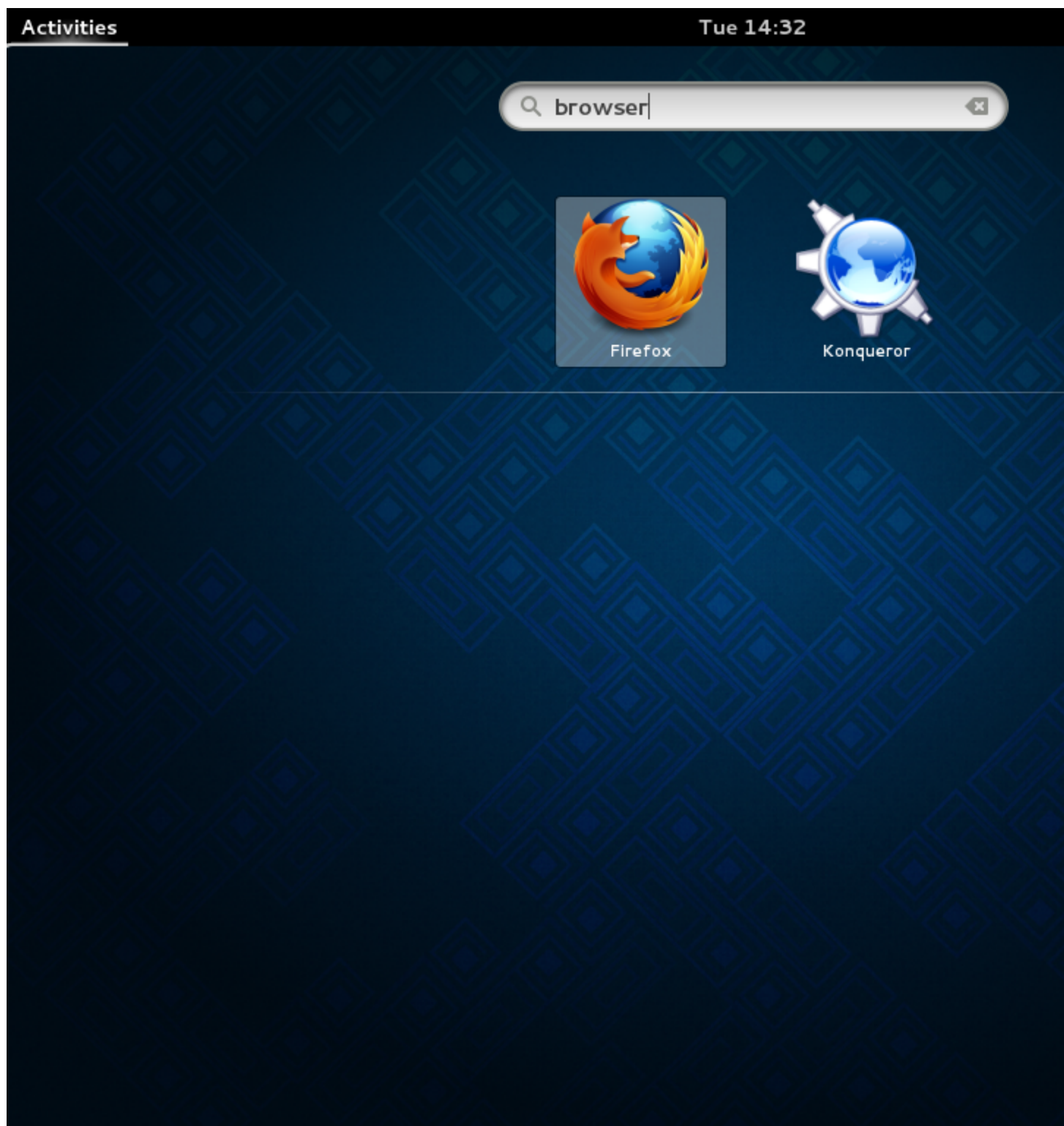


Figure 1.6. Using the GNOME search box

The **overview** can also be browsed. The bar on the left, called the **dash**, shows frequently used applications and a grid icon. Clicking on the grid icon brings up a grid in the center of the window that displays frequently used applications. The grid will display all available applications if selected using the **All** button at the bottom of the screen.



Figure 1.7. Browsing GNOME menu entries

To learn more about using **GNOME shell**, visit <https://wiki.gnome.org/GnomeShell/CheatSheet>

1.3.2. Using KDE menus

The KDE menu is opened by clicking the Fedora button at the bottom left corner of the screen. The menu initially displays favorite applications, which can be added to by right clicking any menu entry.

Hovering over the icons in the lower portion of the menu will display applications, file systems, recently used applications, or options for logging out of the system.

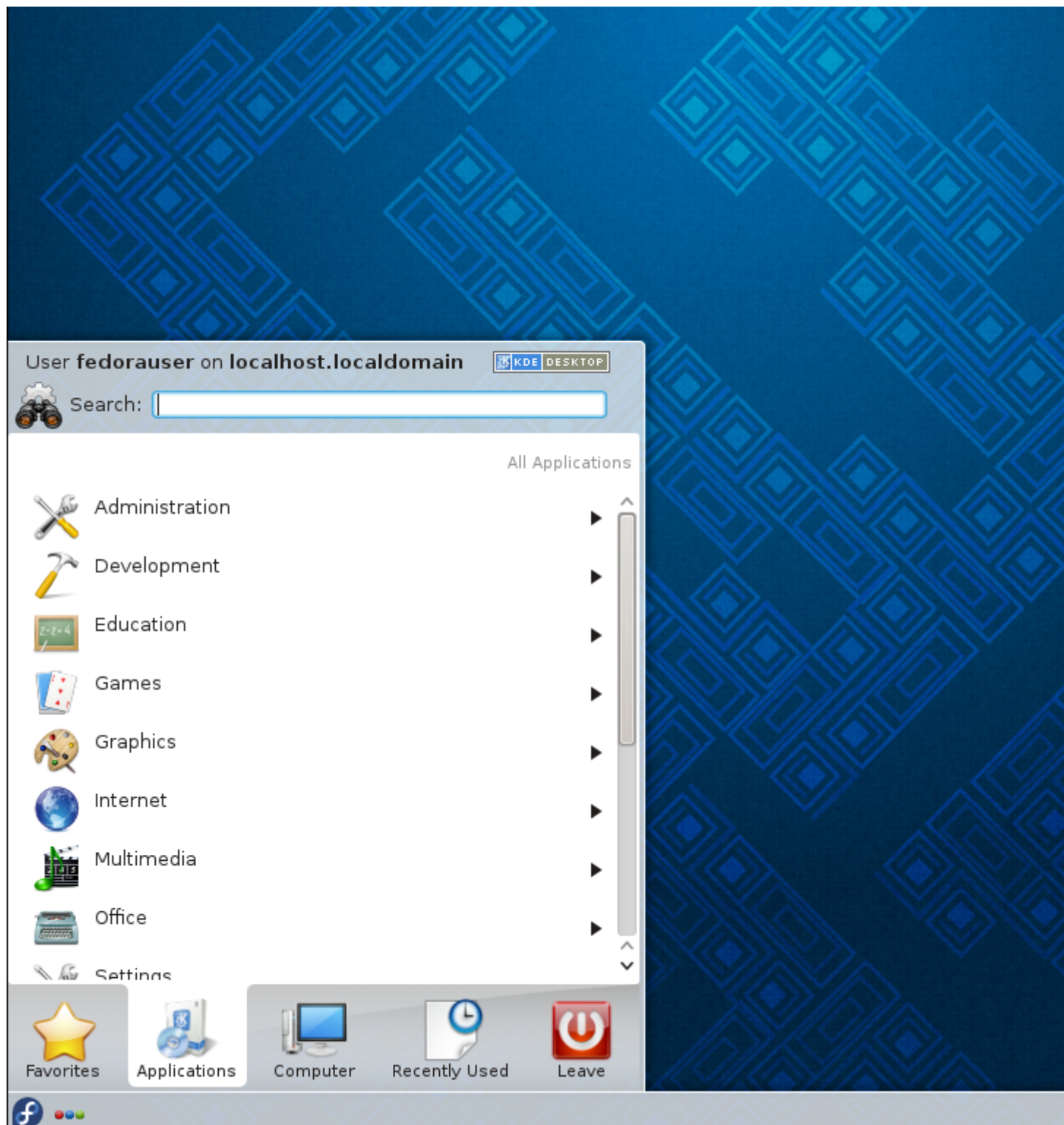


Figure 1.8. The KDE desktop menu.

Search functionality is also available in the KDE menu system. To search for applications, open the menu and begin typing. The menu will display matching entries.

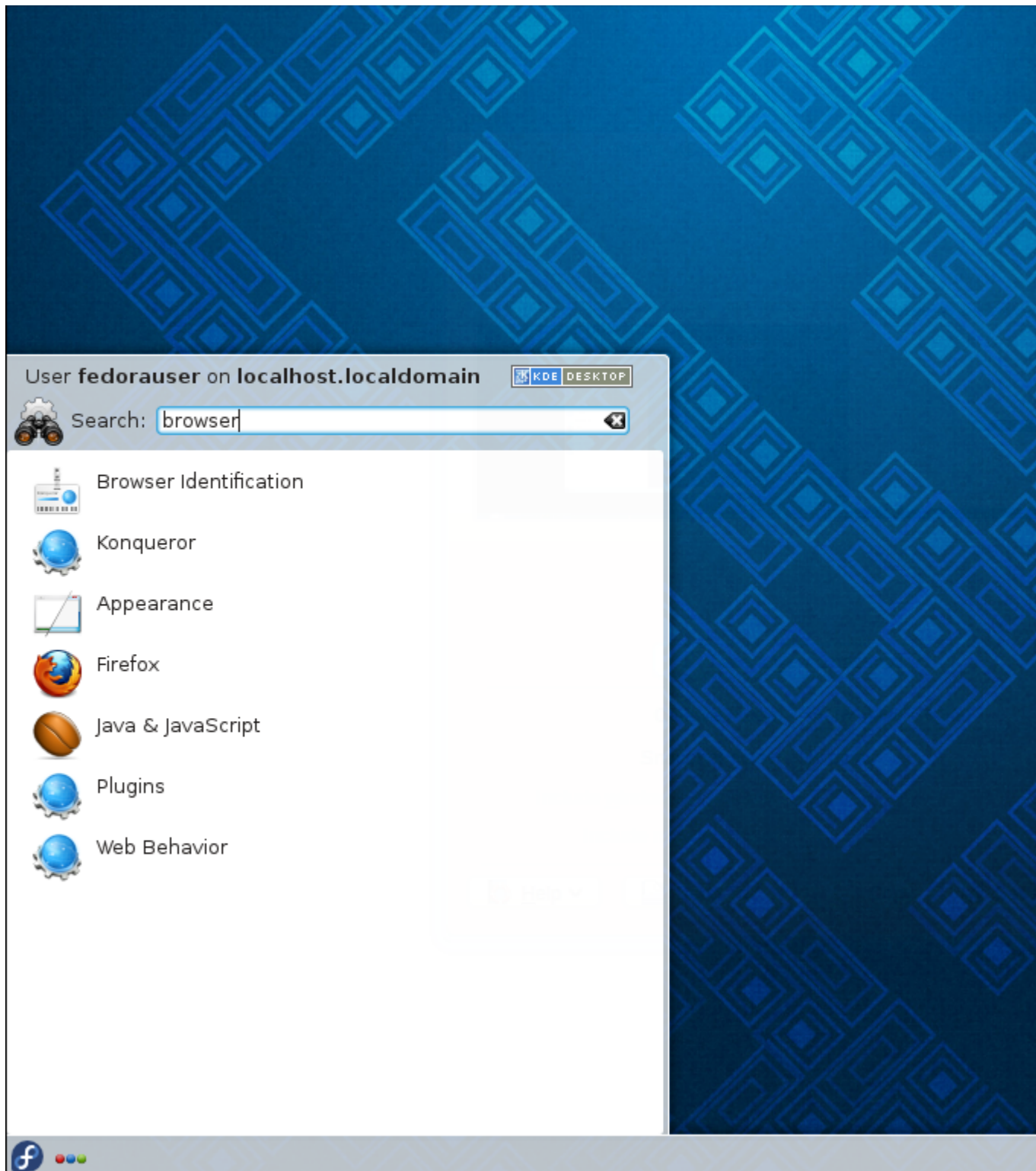


Figure 1.9. Searching with the KDE menu.

1.3.3. Using menus in LXDE, MATE, and XFCE

Menus in LXDE, MATE, and XFCE have a varied appearance but a very similar structure. They categorize applications, and the contents of a category are displayed by hovering the cursor over the entry. Applications are launched by clicking on an entry.

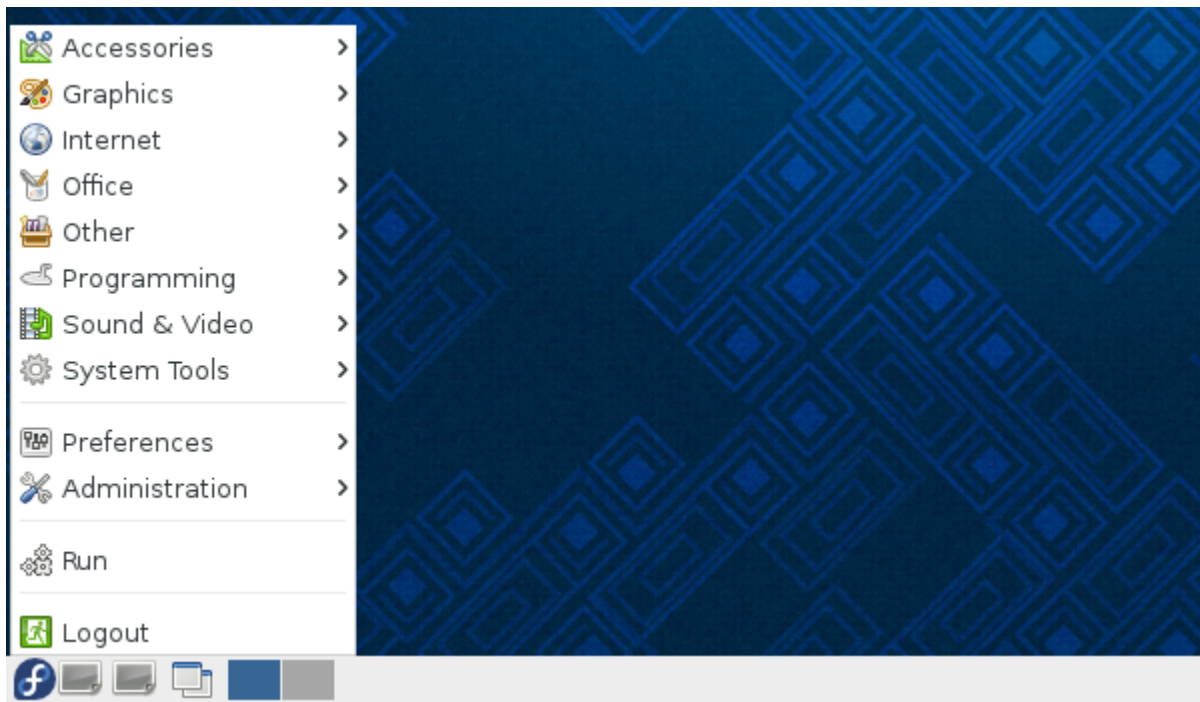


Figure 1.10. The LXDE menu

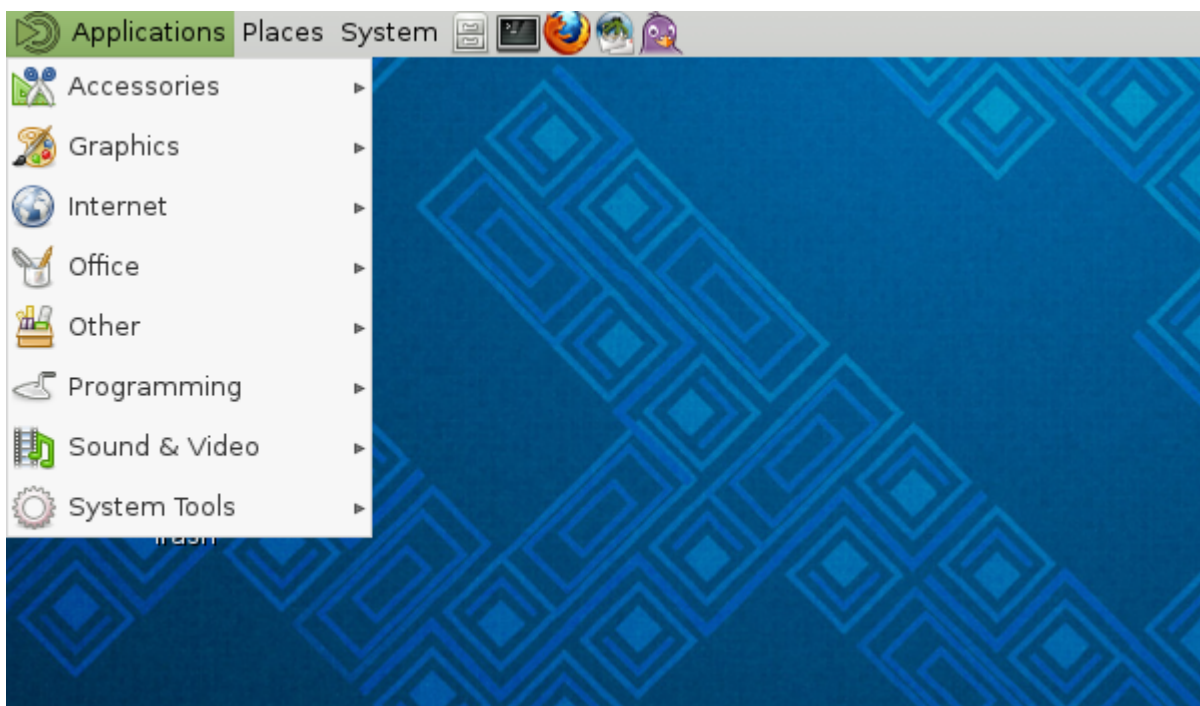


Figure 1.11. MATE menu

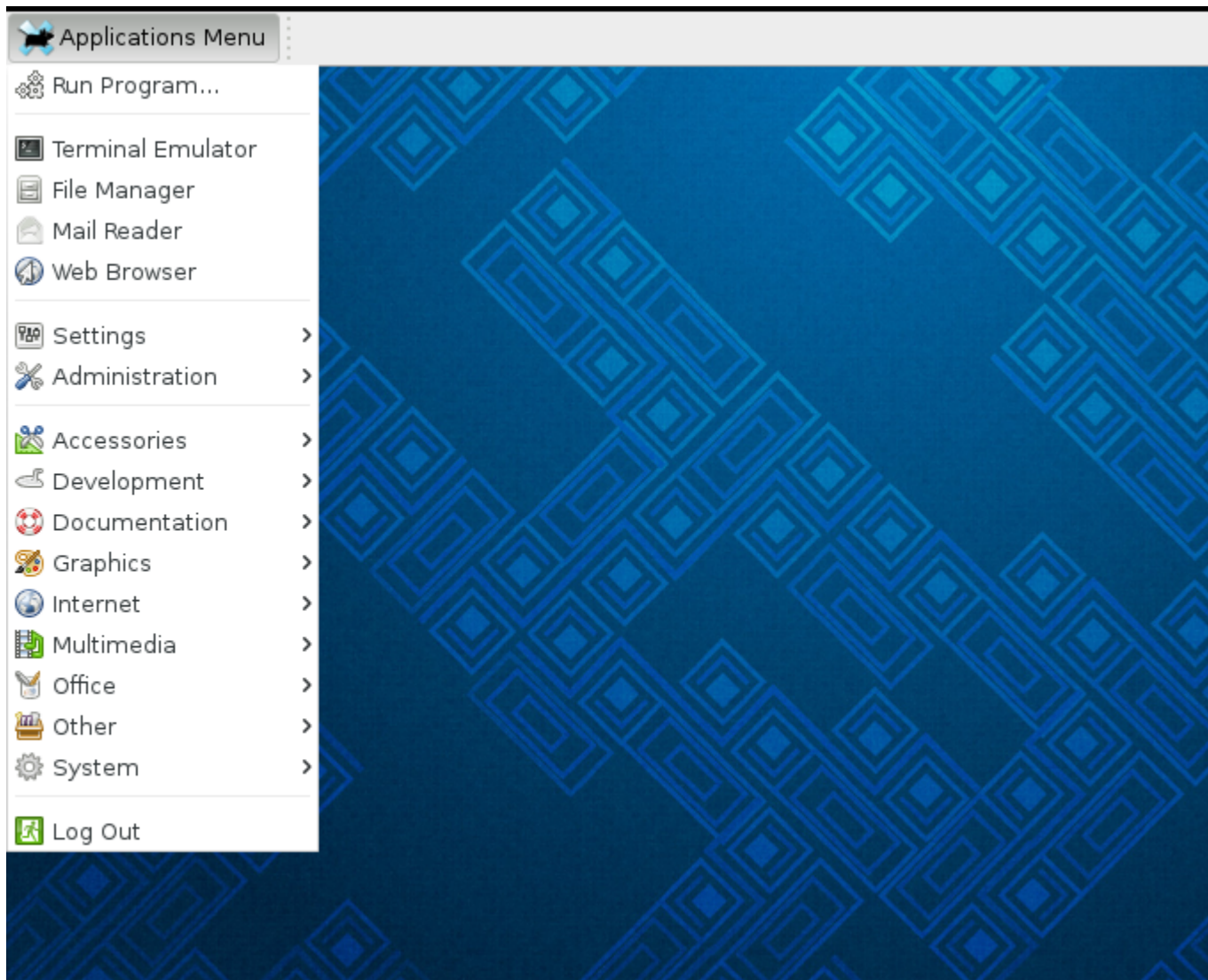


Figure 1.12. XFCE Menu

System Locale and Keyboard Configuration

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

These settings can be made by modifying the `/etc/locale.conf` configuration file or by using the **localectl** utility. Also, you can use the graphical user interface to perform the task; for a description of this method, see [Fedora Installation Guide](#)¹.

2.1. Setting the System Locale

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

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

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

Here, the `LC_MESSAGES` option determines the locale used for diagnostic messages written to the standard error output. To further specify locale settings in `/etc/locale.conf`, you can use several other options, the most relevant are summarized in [Table 2.1, “Options configurable in /etc/locale.conf”](#). See the `locale(7)` manual page for detailed information on these options. Note that the `LC_ALL` option, which represents all possible options, should not be configured in `/etc/locale.conf`.

Table 2.1. Options configurable in `/etc/locale.conf`

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

¹ <http://docs.fedoraproject.org/install-guide>

2.1.1. Displaying the Current Status

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

```
localectl status
```

Example 2.1. Displaying the Current Status

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

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

2.1.2. Listing Available Locales

To list all locales available for your system, type:

```
localectl list-locales
```

Example 2.2. Listing Locales

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

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

output truncated
```

2.1.3. Setting the Locale

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

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

Replace *locale* with the locale name, found with the **localectl list-locales** command. The above syntax can also be used to configure parameters from [Table 2.1, “Options configurable in /etc/locale.conf”](#).

Example 2.3. Changing the Default Locale

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


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

2.2. Changing the Keyboard Layout

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

2.2.1. Displaying the Current Settings

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

```
localectl status
```

Example 2.4. Displaying the Keyboard Settings

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

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

2.2.2. Listing Available Keymaps

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

```
localectl list-keymaps
```

Example 2.5. Searching for a Particular Keymap

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

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

2.2.3. Setting the Keymap

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


```
localectl set-keymap map
```

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

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

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

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

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

Example 2.6. Setting the X11 Keymap Separately

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

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

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

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

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

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

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

2.3. Additional Resources

For more information on how to configure the keyboard layout on Fedora, see the resources listed below:

Installed Documentation

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

Configuring the Date and Time

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

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

The system time is always kept in *Coordinated Universal Time* (UTC) and converted in applications to local time as needed. *Local time* is the actual time in your current time zone, taking into account *daylight saving time* (DST). The real-time clock can use either UTC or local time. UTC is recommended.

Fedora 26 offers three command line tools that can be used to configure and display information about the system date and time: the **timedatectl** utility, which is new in Fedora 26 and is part of `systemd`; the traditional **date** command; and the **hwclock** utility for accessing the hardware clock.

3.1. Using the **timedatectl** Command

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

For information on how to display the current date and time in a custom format, see also [Section 3.2, “Using the `date` Command”](#).

3.1.1. Displaying the Current Date and Time

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

```
timedatectl
```

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

Example 3.1. Displaying the Current Date and Time

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

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



```
Sun 2013-03-31 03:00:00 CEST
Next DST change: DST ends (the clock jumps one hour backwards) at
Sun 2013-10-27 02:59:59 CEST
Sun 2013-10-27 02:00:00 CET
```

3.1.2. Changing the Current Time

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

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

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

This command updates both the system time and the hardware clock. The result it is similar to using both the **date --set** and **hwclock --systohc** commands.

The command will fail if an NTP service is enabled. See [Section 3.1.5, “Synchronizing the System Clock with a Remote Server”](#) to temporally disable the service.

Example 3.2. Changing the Current Time

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

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

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

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

To configure your system to maintain the clock in the local time, replace *boolean* with **yes** (or, alternatively, **y**, **true**, **t**, or **1**). To configure the system to use UTC, replace *boolean* with **no** (or, alternatively, **n**, **false**, **f**, or **0**). The default option is **no**.

3.1.3. Changing the Current Date

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

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

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

Note that changing the date without specifying the current time results in setting the time to 00:00:00.

Example 3.3. Changing the Current Date

To change the current date to 2 June 2013 and keep the current time (11:26 p.m.), run the following command as root:

```
~]# timedatectl set-time "2013-06-02 23:26:00"
```

3.1.4. Changing the Time Zone

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


```
timedatectl list-timezones
```

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

```
timedatectl set-timezone time_zone
```

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

Example 3.4. Changing the Time Zone

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

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

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

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

3.1.5. Synchronizing the System Clock with a Remote Server

As opposed to the manual adjustments described in the previous sections, the **timedatectl** command also allows you to enable automatic synchronization of your system clock with a group of remote servers using the NTP protocol. Enabling NTP enables the **chronyd** or **ntpd** service, depending on which of them is installed.

The NTP service can be enabled and disabled using a command as follows:

```
timedatectl set-ntp boolean
```

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

Example 3.5. Synchronizing the System Clock with a Remote Server

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

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

The command will fail if an NTP service is not installed. See [Section 14.3.1, “Installing chrony”](#) for more information.

3.2. Using the date Command

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

For information on how to change the time zone or enable automatic synchronization of the system clock with a remote server, see [Section 3.1, “Using the `timedatectl` Command”](#).

3.2.1. Displaying the Current Date and Time

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

```
date
```

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

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

```
date --utc
```

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

```
date +"format"
```

Replace *format* with one or more supported control sequences as illustrated in [Example 3.6, “Displaying the Current Date and Time”](#). See [Table 3.1, “Commonly Used Control Sequences”](#) for a list of the most frequently used formatting options, or the `date(1)` manual page for a complete list of these options.

Table 3.1. Commonly Used Control Sequences

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

Example 3.6. Displaying the Current Date and Time

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

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

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

```
~]$ date --utc
```



```
Mon Sep 16 15:30:34 UTC 2013
```

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

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

3.2.2. Changing the Current Time

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

```
date --set HH:MM:SS
```

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

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

```
date --set HH:MM:SS --utc
```

Example 3.7. Changing the Current Time

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

```
~]# date --set 23:26:00
```

3.2.3. Changing the Current Date

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

```
date --set YYYY-MM-DD
```

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

Note that changing the date without specifying the current time results in setting the time to 00:00:00.

Example 3.8. Changing the Current Date

To change the current date to 2 June 2013 and keep the current time (11:26 p.m.), run the following command as root:

```
~]# date --set 2013-06-02 23:26:00
```

3.3. Using the hwclock Command

hwclock is a utility for accessing the hardware clock, also referred to as the Real Time Clock (RTC). The hardware clock is independent of the operating system you use and works even when the machine is shut down. This utility is used for displaying the time from the hardware clock. **hwclock** also contains facilities for compensating for systematic drift in the hardware clock.

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

The `hwclock` utility saves its settings in the `/etc/adjtime` file, which is created with the first change you make, for example, when you set the time manually or synchronize the hardware clock with the system time.

Note

In Fedora 6, the `hwclock` command was run automatically on every system shutdown or reboot, but it is not in Fedora 26. When the system clock is synchronized by the Network Time Protocol (NTP) or Precision Time Protocol (PTP), the kernel automatically synchronizes the hardware clock to the system clock every 11 minutes.

For details about NTP, see [Chapter 14, Configuring NTP Using the chrony Suite](#) and [Chapter 15, Configuring NTP Using ntpd](#). For information about PTP, see [Chapter 16, Configuring PTP Using ptp4l](#). For information about setting the hardware clock after executing `ntpdate`, see [Section 15.18, “Configuring the Hardware Clock Update”](#).

3.3.1. Displaying the Current Date and Time

Running `hwclock` with no command line options as the `root` user returns the date and time in local time to standard output.

```
hwclock
```

Note that using the `--utc` or `--localtime` options with the `hwclock` command does not mean you are displaying the hardware clock time in UTC or local time. These options are used for setting the hardware clock to keep time in either of them. The time is always displayed in local time. Additionally, using the `hwclock --utc` or `hwclock --local` commands does not change the record in the `/etc/adjtime` file. This command can be useful when you know that the setting saved in `/etc/adjtime` is incorrect but you do not want to change the setting. On the other hand, you may receive misleading information if you use the command an incorrect way. See the `hwclock(8)` manual page for more details.

Example 3.9. Displaying the Current Date and Time

To display the current date and the current local time from the hardware clock, run as `root`:

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

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

For information on how to change the time zone, see [Section 3.1.4, “Changing the Time Zone”](#).

3.3.2. Setting the Date and Time

Besides displaying the date and time, you can manually set the hardware clock to a specific time.

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

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


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

At the same time, you can also set the hardware clock to keep the time in either UTC or local time by adding the `--utc` or `--localtime` options, respectively. In this case, **UTC** or **LOCAL** is recorded in the `/etc/adjtime` file.

Example 3.10. Setting the Hardware Clock to a Specific Date and Time

If you want to set the date and time to a specific value, for example, to "21:17, October 21, 2014", and keep the hardware clock in UTC, run the command as root in the following format:

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

3.3.3. Synchronizing the Date and Time

You can synchronize the hardware clock and the current system time in both directions.

- Either you can set the hardware clock to the current system time by using this command:

```
hwclock --systohc
```

Note that if you use NTP, the hardware clock is automatically synchronized to the system clock every 11 minutes, and this command is useful only at boot time to get a reasonable initial system time.

- Or, you can set the system time from the hardware clock by using the following command:

```
hwclock --hctosys
```

When you synchronize the hardware clock and the system time, you can also specify whether you want to keep the hardware clock in local time or UTC by adding the `--utc` or `--localtime` option. Similarly to using `--set`, **UTC** or **LOCAL** is recorded in the `/etc/adjtime` file.

The `hwclock --systohc --utc` command is functionally similar to `timedatectl set-local-rtc false` and the `hwclock --systohc --localtime` command is an alternative to `timedatectl set-local-rtc true`.

Example 3.11. Synchronizing the Hardware Clock with System Time

To set the hardware clock to the current system time and keep the hardware clock in local time, run the following command as root:

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

To avoid problems with time zone and DST switching, it is recommended to keep the hardware clock in UTC. The shown [Example 3.11, "Synchronizing the Hardware Clock with System Time"](#) is useful, for example, in case of a multi boot with a Windows system, which assumes the hardware clock runs in local time by default, and all other systems need to accommodate to it by using local time as well. It may also be needed with a virtual machine; if the virtual hardware clock provided by the host is running in local time, the guest system needs to be configured to use local time, too.

3.4. Additional Resources

For more information on how to configure the date and time in Fedora 26, see the resources listed below.

Installed Documentation

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

See Also

- [Chapter 2, System Locale and Keyboard Configuration](#) documents how to configure the keyboard layout.

Managing Users and Groups

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

4.1. Introduction to Users and Groups

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

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

Additionally, Fedora supports *access control lists* (ACLs) for files and directories which allow permissions for specific users outside of the owner to be set. For more information about this feature, see the [Access Control Lists](https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Storage_Administration_Guide/ch-acls.html)¹ chapter of the [Storage Administration Guide](https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Storage_Administration_Guide/index.html)².

4.1.1. User Private Groups

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

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

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

A list of all groups is stored in the `/etc/group` configuration file.

4.1.2. Shadow Passwords

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

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

¹ https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Storage_Administration_Guide/ch-acls.html

² https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Storage_Administration_Guide/index.html

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

Most utilities provided by the `shadow-utils` package work properly whether or not shadow passwords are enabled. However, since password aging information is stored exclusively in the `/etc/shadow` file, some utilities and commands do not work without first enabling shadow passwords:

- The **chage** utility for setting password-aging parameters. For details, see the [Password Security](#)³ section in the *Red Hat Enterprise Linux 7 Security Guide*.
- The **gpasswd** utility for administrating the `/etc/group` file.
- The **usermod** command with the `-e`, `--expiredate` or `-f`, `--inactive` option.
- The **useradd** command with the `-e`, `--expiredate` or `-f`, `--inactive` option.

4.2. Managing Users in a Graphical Environment

The **Users** utility allows you to view, modify, add, and delete local users in the graphical user interface.

4.2.1. Using the Users Settings Tool

Press the **Super** key to enter the Activities Overview, type **Users** and then press **Enter**. The **Users** settings tool appears. The **Super** key appears in a variety of guises, depending on the keyboard and other hardware, but often as either the Windows or Command key, and typically to the left of the Spacebar.

To make changes to the user accounts, first select the **Unlock** button and authenticate yourself as indicated by the dialog box that appears. Note that unless you have superuser privileges, the application will prompt you to authenticate as root. To add and remove users, select the **+** and **-** button respectively. To add a user to the administrative group `wheel`, change the **Account Type** from **Standard** to **Administrator**. To edit a user's language setting, select the language and a drop-down menu appears.

³ https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/chap-Hardening_Your_System_with_Tools_and_Services.html#sec-Password_Security

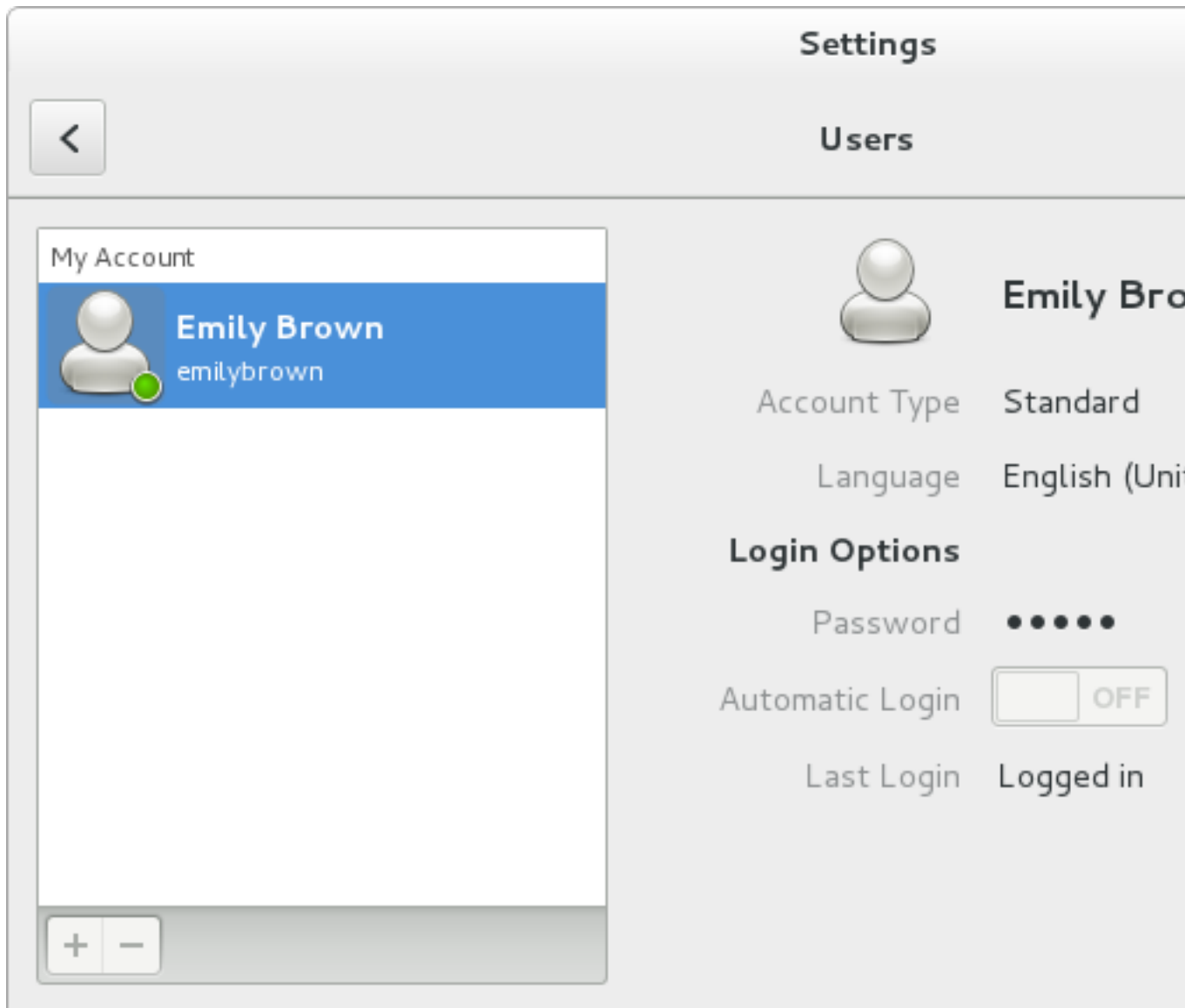


Figure 4.1. The Users Settings Tool

When a new user is created, the account is disabled until a password is set. The **Add User** menu contains the options to set a password by the administrator immediately, or to allow the user to choose a password at the first login.

4.3. Using Command Line Tools

Apart from the **Users** settings tool described in [Section 4.2, “Managing Users in a Graphical Environment”](#), which is designed for basic managing of users, you can use command line tools for managing users and groups that are listed in [Table 4.1, “Command line utilities for managing users and groups”](#).

Table 4.1. Command line utilities for managing users and groups

Utilities	Description
id	Displays user and group IDs.
useradd, usermod, userdel	Standard utilities for adding, modifying, and deleting user accounts.

Utilities	Description
groupadd, groupmod, groupdel	Standard utilities for adding, modifying, and deleting groups.
passwd	Standard utility for administering the /etc/group configuration file.
pwck, grpck	Utilities that can be used for verification of the password, group, and associated shadow files.
pwconv, pwunconv	Utilities that can be used for the conversion of passwords to shadow passwords, or back from shadow passwords to standard passwords.
grpconv, grpunconv	Similar to the previous, these utilities can be used for conversion of shadowed information for group accounts.

4.3.1. Adding a New User

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

```
useradd [options] username
```

...where *options* are command-line options as described in [Table 4.2, “Common useradd command-line options”](#).

By default, the **useradd** command creates a locked user account. To unlock the account, run the following command as root to assign a password:

```
passwd username
```

Optionally, you can set a password aging policy. See [Section 4.3.3, “Enabling Password Aging”](#) for information on how to enable password aging.

Table 4.2. Common useradd command-line options

Option	Description
-c 'comment'	<i>comment</i> can be replaced with any string. This option is generally used to specify the full name of a user.
-d home_directory	Home directory to be used instead of default /home/username/ .
-e date	Date for the account to be disabled in the format YYYY-MM-DD.
-f days	Number of days after the password expires until the account is disabled. If 0 is specified, the account is disabled immediately after the password expires. If -1 is specified, the account is not disabled after the password expires.
-g group_name	Group name or group number for the user's default (primary) group. The group must exist prior to being specified here.
-G group_list	List of additional (supplementary, other than default) group names or group numbers, separated by commas, of which the user is a member. The groups must exist prior to being specified here.
-m	Create the home directory if it does not exist.

Option	Description
-M	Do not create the home directory.
-N	Do not create a user private group for the user.
-p password	The password encrypted with crypt .
-r	Create a system account with a UID less than 1000 and without a home directory.
-s	User's login shell, which defaults to /bin/bash .
-u uid	User ID for the user, which must be unique and greater than 999.

The command-line options associated with the **usermod** command are essentially the same. Note that if you want to add a user to another supplementary group, you need to use the **-a**, **--append** option with the **-G** option. Otherwise the list of supplementary groups for the user will be overwritten by those specified with the **usermod -G** command.

Explaining the Process

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

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

```
juan:x:1001:1001::/home/juan:/bin/bash
```

The line has the following characteristics:

- It begins with the user name **juan**.
- There is an **x** for the password field indicating that the system is using shadow passwords.
- A UID greater than 999 is created. Under Fedora, UIDs below 1000 are reserved for system use and should not be assigned to users.
- A GID greater than 999 is created. Under Fedora, GIDs below 1000 are reserved for system use and should not be assigned to users.
- The optional **GECOS** information is left blank. The **GECOS** field can be used to provide additional information about the user, such as their full name or phone number.
- The home directory for **juan** is set to **/home/juan/**.
- The default shell is set to **/bin/bash**.

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

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

The line has the following characteristics:

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



Note

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

- The password is set to never expire.

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

```
juan:x:1001:
```

A group with the same name as a user is called a *user private group*. For more information on user private groups, see [Section 4.1.1, “User Private Groups”](#).

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

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

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

```
juan:::
```

The line has the following characteristics:

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

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

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

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

6. The files within the **/etc/skel/** directory (which contain default user settings) are copied into the new **/home/juan/** directory. The contents of **/etc/skel/** may vary depending on installed applications:

```
~]# ls -la /home/juan
total 24
drwx-----. 4 juan juan 4096 Mar  3 18:23 .
```



```
drwxr-xr-x. 5 root root 4096 Mar  3 18:23 ..
-rw-r--r--. 1 juan juan  18 Jul 09 08:43 .bash_logout
-rw-r--r--. 1 juan juan 176 Jul 09 08:43 .bash_profile
-rw-r--r--. 1 juan juan 124 Jul 09 08:43 .bashrc
drwxr-xr-x. 4 juan juan 4096 Jul 09 08:43 .mozilla
```

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

4.3.2. Adding a New Group

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

```
groupadd [options] group_name
```

...where *options* are command-line options as described in [Table 4.3, “Common groupadd command-line options”](#).

Table 4.3. Common groupadd command-line options

Option	Description
-f, --force	When used with -g gid and <i>gid</i> already exists, groupadd will choose another unique <i>gid</i> for the group.
-g gid	Group ID for the group, which must be unique and greater than 999.
-K, --key key=value	Override <code>/etc/login.defs</code> defaults.
-o, --non-unique	Allows creating groups with duplicate GID.
-p, --password password	Use this encrypted password for the new group.
-r	Create a system group with a GID less than 1000.

4.3.3. Enabling Password Aging

For security reasons, it is advisable to require users to change their passwords periodically. This can be done by using the **chage** command.



Shadow passwords must be enabled to use chage

Shadow passwords must be enabled to use the **chage** command. For more information, see [Section 4.1.2, “Shadow Passwords”](#).

To configure password expiration for a user from a shell prompt, run the following command as `root`:

```
chage [options] username
```

...where *options* are command line options as described in [Table 4.4, “chage command line options”](#). When the **chage** command is followed directly by a username (that is, when no command

line options are specified), it displays the current password aging values and allows you to change them interactively.

Table 4.4. `chage` command line options

Option	Description
-d <i>days</i>	Specifies the number of days since January 1, 1970 the password was changed.
-E <i>date</i>	Specifies the date on which the account is locked, in the format YYYY-MM-DD. Instead of the date, the number of days since January 1, 1970 can also be used.
-I <i>days</i>	Specifies the number of inactive days after the password expiration before locking the account. If the value is 0 , the account is not locked after the password expires.
-l	Lists current account aging settings.
-m <i>days</i>	Specify the minimum number of days after which the user must change passwords. If the value is 0 , the password does not expire.
-M <i>days</i>	Specify the maximum number of days for which the password is valid. When the number of days specified by this option plus the number of days specified with the -d option is less than the current day, the user must change passwords before using the account.
-W <i>days</i>	Specifies the number of days before the password expiration date to warn the user.

You can configure a password to expire the first time a user logs in. This forces users to change passwords immediately.

1. Set up an initial password. There are two common approaches to this step: you can either assign a default password, or you can use a null password.

To assign a default password, type the following at a shell prompt as root:

```
passwd username
```

To assign a null password instead, use the following command:

```
passwd -d username
```



Avoid using null passwords whenever possible

Using a null password, while convenient, is a highly insecure practice, as any third party can log in first and access the system using the insecure username. Always make sure that the user is ready to log in before unlocking an account with a null password.

2. Force immediate password expiration by running the following command as root:

```
chage -d 0 username
```


This command sets the value for the date the password was last changed to the epoch (January 1, 1970). This value forces immediate password expiration no matter what password aging policy, if any, is in place.

Upon the initial log in, the user is now prompted for a new password.

4.3.4. Enabling Automatic Logouts

Especially when the user is logged in as **root**, an unattended login session may pose a significant security risk. To reduce this risk, you can configure the system to automatically log out idle users after a fixed period of time:

1. Make sure the **screen** package is installed. You can do so by running the following command as **root**:

```
dnf install screen
```

For more information on how to install packages in Fedora, refer to [Section 6.2.4, “Installing Packages”](#).

2. As **root**, add the following line at the beginning of the **/etc/profile** file to make sure the processing of this file cannot be interrupted:

```
trap "" 1 2 3 15
```

3. Add the following lines at the end of the **/etc/profile** file to start a **screen** session each time a user logs in to a virtual console or remotely:

```
SCREENEXEC="screen"
if [ -w $(tty) ]; then
    trap "exec $SCREENEXEC" 1 2 3 15
    echo -n 'Starting session in 10 seconds'
    sleep 10
    exec $SCREENEXEC
fi
```

Note that each time a new session starts, a message will be displayed and the user will have to wait ten seconds. To adjust the time to wait before starting a session, change the value after the **sleep** command.

4. Add the following lines to the **/etc/screenrc** configuration file to close the **screen** session after a given period of inactivity:

```
idle 120 quit
autodetach off
```

This will set the time limit to 120 seconds. To adjust this limit, change the value after the **idle** directive.

Alternatively, you can configure the system to only lock the session by using the following lines instead:

```
idle 120 lockscreen
autodetach off
```


This way, a password will be required to unlock the session.

The changes take effect the next time a user logs in to the system.

4.3.5. Creating Group Directories

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

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

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

```
mkdir /opt/myproject
```

2. Add the myproject group to the system:

```
groupadd myproject
```

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

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

4. Allow users in the group to create files within the directory and set the *setgid* bit:

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

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

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

5. Add users to the myproject group:

```
usermod -aG myproject username
```

4.4. Additional Resources

For more information on how to manage users and groups on Fedora, see the resources listed below.

Installed Documentation

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

- `useradd(8)` — The manual page for the **useradd** command documents how to use it to create new users.
- `userdel(8)` — The manual page for the **userdel** command documents how to use it to delete users.
- `usermod(8)` — The manual page for the **usermod** command documents how to use it to modify users.
- `groupadd(8)` — The manual page for the **groupadd** command documents how to use it to create new groups.
- `groupdel(8)` — The manual page for the **groupdel** command documents how to use it to delete groups.
- `groupmod(8)` — The manual page for the **groupmod** command documents how to use it to modify group membership.
- `gpasswd(1)` — The manual page for the **gpasswd** command documents how to manage the **/etc/group** file.
- `grpck(8)` — The manual page for the **grpck** command documents how to use it to verify the integrity of the **/etc/group** file.
- `pwck(8)` — The manual page for the **pwck** command documents how to use it to verify the integrity of the **/etc/passwd** and **/etc/shadow** files.
- `pwconv(8)` — The manual page for the **pwconv**, **pwunconv**, **grpconv**, and **grpunconv** commands documents how to convert shadowed information for passwords and groups.
- `id(1)` — The manual page for the **id** command documents how to display user and group IDs.

For information about related configuration files, see:

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

Gaining Privileges

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

See the [Red Hat Enterprise Linux 7 Security Guide](#)¹ for more information on administrative controls, potential dangers, and ways to prevent data loss resulting from improper use of privileged access.

5.1. The su Command

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

Once logged in using the **su** command, the user **is** the root user and has absolute administrative access to the system. Note that this access is still subject to the restrictions imposed by SELinux, if it is enabled. In addition, once a user has become root, it is possible for them to use the **su** command to change to any other user on the system without being prompted for a password.

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

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

```
~]# usermod -a -G wheel username
```

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

You can also use the **Users** settings tool to modify group memberships, as follows. Note that you need administrator privileges to perform this procedure.

1. Press the **Super** key to enter the Activities Overview, type **Users** and then press **Enter**. The **Users** settings tool appears. The **Super** key appears in a variety of guises, depending on the keyboard and other hardware, but often as either the Windows or Command key, and typically to the left of the **Spacebar**.
2. To enable making changes, click the **Unlock** button, and enter a valid administrator password.
3. Click a user icon in the left column to display the user's properties in the right-hand pane.
4. Change the **Account Type** from **Standard** to **Administrator**. This will add the user to the *wheel* group.

See [Section 4.2, "Managing Users in a Graphical Environment"](#) for more information about the **Users** tool.

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

¹ https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/


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

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

Note

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

5.2. The sudo Command

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

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

```
sudo command
```

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

The **sudo** command allows for a high degree of flexibility. For instance, only users listed in the `/etc/sudoers` configuration file are allowed to use the **sudo** command and the command is executed in *the user's* shell, not a root shell. This means the root shell can be completely disabled as shown in the [Red Hat Enterprise Linux 7 Security Guide](https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/)².

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

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

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

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

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

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

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

² https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/


```
juaan ALL=(ALL) ALL
```

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

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

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

This example states that any member of the user's system group can issue the command **/sbin/shutdown -h now** as long as it is issued from the console.

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



Important

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

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

```
Defaults    timestamp_timeout=value
```

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

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

```
sudo /bin/bash
```

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

5.3. Additional Resources

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

Installed Documentation

- **su(1)** — The manual page for **su** provides information regarding the options available with this command.

- `sudo(8)` — The manual page for **sudo** includes a detailed description of this command and lists options available for customizing its behavior.
- `pam(8)` — The manual page describing the use of Pluggable Authentication Modules (PAM) for Linux.

Online Documentation

- The [Red Hat Enterprise Linux 7 Security Guide](https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/)³ provides a more in-depth look at potential security issues pertaining to `setuid` programs as well as techniques used to alleviate these risks.

See Also

- [Chapter 4, Managing Users and Groups](#) documents how to manage system users and groups in the graphical user interface and on the command line.

³ https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/

Part II. Package Management

All software on a Fedora system is divided into RPM packages, which can be installed, upgraded, or removed. This part describes how to manage packages on Fedora using **DNF**.

DNF

DNF is the The Fedora Project package manager that is able to query for information about packages, fetch packages from repositories, install and uninstall packages using automatic dependency resolution, and update an entire system to the latest available packages. DNF performs automatic dependency resolution on packages you are updating, installing or removing, and thus is able to automatically determine, fetch and install all available dependent packages. DNF can be configured with new, additional repositories, or *package sources*, and also provides many plug-ins which enhance and extend its capabilities. DNF is able to perform many of the same tasks that **RPM** can; additionally, many of the command line options are similar. DNF enables easy and simple package management on a single machine or on groups of them.



Secure package management with GPG-signed packages

DNF provides secure package management by enabling GPG (Gnu Privacy Guard; also known as GnuPG) signature verification on GPG-signed packages to be turned on for all package repositories (package sources), or for individual repositories. When signature verification is enabled, DNF will refuse to install any packages not GPG-signed with the correct key for that repository. This means that you can trust that the **RPM** packages you download and install on your system are from a trusted source, such as The Fedora Project, and were not modified during transfer. See [Section 6.3, “Configuring DNF and DNF Repositories”](#) for details on enabling signature-checking with DNF, or [Section A.3.2, “Checking Package Signatures”](#) for information on working with and verifying GPG-signed **RPM** packages in general.

DNF also enables you to easily set up your own repositories of **RPM** packages for download and installation on other machines.

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



DNF and superuser privileges

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

6.1. Checking For and Updating Packages

6.1.1. Checking For Updates

The quickest way to check for updates is to attempt to install any available updates by using the **dnf upgrade** command as follows:

```
~]# dnf upgrade
Last metadata expiration check performed 1:24:32 ago on Thu May 14 23:23:51 2015.
```



```
Dependencies resolved.  
Nothing to do.  
Complete!
```

Note that **dnf upgrade** installs only those updates that can be installed. If a package cannot be updated, because of dependency problems for example, it is skipped.

The **dnf check-update** command can be used see which installed packages on your system have new versions available, however it does not mean that they can be successfully installed. This command is therefore mostly useful in scripts and for checking for updated packages that were not installed after running **dnf upgrade**.

For example:

```
~]# dnf check-update  
Using metadata from Mon Apr 20 16:34:10 2015 (2:42:10 hours old)  
  
python.x86_64                2.7.9-6.fc22            updates  
python-cryptography.x86_64  0.8.2-1.fc22            updates  
python-libs.x86_64          2.7.9-6.fc22            updates
```

The packages in the above output are listed as having updated versions. The line in the example output tells us:

- **python** — the name of the package,
- **x86_64** — the CPU architecture the package was built for,
- **2.7.9** — the version of the updated package,
- **6.fc22** — the release of the updated package,
- **updates-testing** — the repository in which the updated package is located.

6.1.2. Updating Packages

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

Updating a Single Package

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

```
dnf upgrade package_name
```

For example, to update the *python* package, type:

```
~]# dnf upgrade python  
Using metadata from Mon Apr 20 16:38:16 2015 (2:42:14 hours old)  
Dependencies resolved.  
=====
```

Package	Arch	Version	Repository	Size
---------	------	---------	------------	------

```
=====
```

Upgrading:


```
python      x86_64      2.7.9-6.fc22 updates      92 k
python-libs x86_64      2.7.9-6.fc22 updates      5.8 M

Transaction Summary
=====
Upgrade 2 Packages

Total download size: 5.9 M
Is this ok [y/N]:
```

This output contains:

1. **python.x86_64** — you can download and install new *python* package.
2. **python-libs.x86_64** — DNF has resolved that the *python-libs-2.7.9-6.fc22.x86_64* package is a required dependency of the *python* package.
3. DNF presents the update information and then prompts you as to whether you want it to perform the update; DNF runs interactively by default. If you already know which transactions DNF plans to perform, you can use the **-y** option to automatically answer **yes** to any questions DNF may ask (in which case it runs non-interactively). However, you should always examine which changes DNF plans to make to the system so that you can easily troubleshoot any problems that might arise.

If a transaction does go awry, you can view DNF's transaction history by using the **dnf history** command as described in [Section 6.2.6, “Working with Transaction History”](#).



Updating and installing kernels with DNF

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

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

Updating All Packages and Their Dependencies

To update all packages and their dependencies, enter **dnf upgrade** without any arguments:

```
dnf upgrade
```

6.1.3. Preserving Configuration File Changes

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

6.2. Packages and Package Groups

6.2.1. Searching Packages

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

```
dnf search term...
```

Add the **all** to match against descriptions and URLs.

```
dnf search all term...
```

This command displays the list of matches for each term. For example, to list all packages that match “meld” or “kompare”, type:

```
~]# dnf search meld kompare
Loaded plugins: langpacks, presto, refresh-packagekit
===== N/S Matched: meld =====
meld.noarch : Visual diff and merge tool
python-meld3.x86_64 : HTML/XML templating system for Python

===== N/S Matched: kompare =====
komparator.x86_64 : Kompare and merge two folders

Name and summary matches only, use "search all" for everything.
```

6.2.2. Listing Packages

dnf list and related commands provide information about packages, package groups, and repositories.

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

Filtering results with glob expressions

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

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

DNF searches only package names when using glob expressions. To search for a version of a package, include a dash and part of the version number as follows:

```
~]# dnf search kernel*-4*
Last metadata expiration check performed 2:46:09 ago on Thu May 14 23:23:51 2015.
Installed Packages
kernel.x86_64                4.0.0-1.fc22                @System
kernel.x86_64                4.0.2-300.fc22              @System
kernel-core.x86_64           4.0.0-1.fc22                @System
kernel-core.x86_64           4.0.2-300.fc22              @System
[output truncated]
```

See [Example 6.1, “Listing all ABRT addons and plug-ins using glob expressions”](#) and [Example 6.4, “Listing available packages using a single glob expression with escaped wildcard characters”](#) for an example usage of both these methods.

dnf list glob_expression...

Lists information on installed and available packages matching all glob expressions.

Example 6.1. Listing all ABRT addons and plug-ins using glob expressions

Packages with various ABRT addons and plug-ins either begin with “abrt-addon-”, or “abrt-plugin-”. To list these packages, type the following at a shell prompt:

```
~]# dnf list abrt-addon\* abrt-plugin\*
Last metadata expiration check performed 0:14:36 ago on Mon May 25 23:38:13 2015.
Installed Packages
abrt-addon-ccpp.x86_64        2.5.1-2.fc22                @System
abrt-addon-coredump-helper.x86_64 2.5.1-2.fc22                @System
abrt-addon-kerneloops.x86_64    2.5.1-2.fc22                @System
abrt-addon-pstoreoops.x86_64    2.5.1-2.fc22                @System
abrt-addon-python.x86_64        2.5.1-2.fc22                @System
abrt-addon-python3.x86_64       2.5.1-2.fc22                @System
abrt-addon-vmcore.x86_64        2.5.1-2.fc22                @System
abrt-addon-xorg.x86_64          2.5.1-2.fc22                @System
abrt-plugin-bodhi.x86_64        2.5.1-2.fc22                @System
Available Packages
abrt-addon-upload-watch.x86_64    2.5.1-2.fc22                fedora
```

dnf list all

Lists all installed *and* available packages.

Example 6.2. Listing all installed and available packages

```
~]# dnf list all
Last metadata expiration check performed 0:21:11 ago on Mon May 25 23:38:13 2015.
Installed Packages
NetworkManager.x86_64                1:1.0.2-1.fc22      @System
NetworkManager-libnm.x86_64          1:1.0.2-1.fc22      @System
PackageKit.x86_64                    1.0.6-4.fc22        @System
PackageKit-glib.x86_64               1.0.6-4.fc22        @System
aajohan-comfortaa-fonts.noarch        2.004-4.fc22        @System
abrt.x86_64                          2.5.1-2.fc22        @System
[output truncated]
```

dnf list installed

Lists all packages installed on your system. The rightmost column in the output lists the repository from which the package was retrieved.

Example 6.3. Listing installed packages using a double-quoted glob expression

To list all installed packages that begin with “krb” followed by exactly one character and a hyphen, type:

```
~]# dnf list installed "krb?-*"
Last metadata expiration check performed 0:34:45 ago on Mon May 25 23:38:13 2015.
Installed Packages
krb5-libs.x86_64                    1.13.1-3.fc22      @System
krb5-workstation.x86_64             1.13.1-3.fc22      @System
```

dnf list available

Lists all available packages in all enabled repositories.

Example 6.4. Listing available packages using a single glob expression with escaped wildcard characters

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

```
~]# dnf list available gstreamer\*plugin\*
Last metadata expiration check performed 0:42:15 ago on Mon May 25 23:38:13 2015.
Available Packages
gstreamer-plugin-crystalhd.i686      3.10.0-8.fc22      fedora
gstreamer-plugin-crystalhd.x86_64    3.10.0-8.fc22      fedora
gstreamer-plugins-bad-free.i686      0.10.23-24.fc22    fedora
gstreamer-plugins-bad-free.x86_64    0.10.23-24.fc22    fedora
gstreamer-plugins-bad-free-devel.i686 0.10.23-24.fc22    fedora
gstreamer-plugins-bad-free-devel.x86_64 0.10.23-24.fc22    fedora
[output truncated]
```

dnf group list

Lists all package groups.

Example 6.5. Listing all package groups

```
~]# dnf group list
Loaded plugins: langpacks, presto, refresh-packagekit
```



```
Setting up Group Process
Installed Groups:
  Administration Tools
  Design Suite
  Dial-up Networking Support
  Fonts
  GNOME Desktop Environment
[output truncated]
```

dnf repolist

Lists the repository ID, name, and number of packages it provides for each *enabled* repository.

Example 6.6. Listing enabled repositories

```
~]# dnf repolist
Last metadata expiration check performed 0:48:29 ago on Mon May 25 23:38:13 2015.
repo id                repo name                status
*fedora                Fedora 22 - x86_64        44,762
*updates               Fedora 22 - x86_64 - Updates 0
```

dnf repository-packages *repo_id* list

Lists the packages from the specified repository.

Example 6.7. Listing packages from a single repository

```
~]# dnf repository-packages fedora list [option]
Last metadata expiration check performed 1:38:25 ago on Wed May 20 22:16:16 2015.
Installed Packages
PackageKit.x86_64                1.0.6-3.fc22                @System
PackageKit-glib.x86_64           1.0.6-3.fc22                @System
aajohan-comfortaa-fonts.noarch    2.004-4.fc22                @System
[output truncated]
```

The default action is to list all packages available and installed from the repository specified. Add the **available** or **installed** option to list only those packages available or installed from the specified repository.

6.2.3. Displaying Package Information

To display information about one or more packages, use a command as follows:

```
dnf info package_name...
```

For example, to display information about the *abrt* package, type:

```
~]# dnf info abrt
Last metadata expiration check: 1:09:44 ago on Tue May 31 06:51:51 2016.
Installed Packages
Name       : abrt
Arch       : x86_64
Epoch     : 0
Version    : 2.8.1
Release    : 1.fc24
Size       : 2.2 M
Repo       : @System
```



```
From repo    : updates-testing
Summary      : Automatic bug detection and reporting tool
URL          : https://abrt.readthedocs.org/
License      : GPLv2+
Description  : abrt is a tool to help users to detect defects in applications and
              : to create a bug report with all information needed by maintainer to fix it.
              : It uses plugin system to extend its functionality.
```

The **dnf info *package_name*** command is similar to the **rpm -q --info *package_name*** command, but provides as additional information the name of the DNF repository the RPM package was installed from (look for the **From repo:** line in the output). The **dnf info** command shows only the newest available package if there is a newer version available than the one installed. The **dnf repoquery** command can show **all** installed and available packages.

To display information about all available packages, both installed and available from a repository, use a command as follows:

```
dnf repoquery package_name --info
```

For example, to display information about the *abrt* package, type:

```
~]# dnf repoquery abrt --info
Last metadata expiration check: 1:01:44 ago on Tue May 31 06:51:51 2016.
Name           : abrt
Version        : 2.8.1
Release        : 1.fc24
Architecture   : x86_64
Size           : 2318452
License        : GPLv2+
Source RPM     : abrt-2.8.1-1.fc24.src.rpm
Build Date     : 2016-05-25 08:54
Packager       : Fedora Project
URL            : https://abrt.readthedocs.org/
Summary        : Automatic bug detection and reporting tool
Description    :
abrt is a tool to help users to detect defects in applications and
to create a bug report with all information needed by maintainer to fix it.
It uses plugin system to extend its functionality.
```

See the **dnf repoquery** usage statement for more options:

```
~]$ dnf repoquery -h
usage: dnf [options] COMMAND
output truncated
```

6.2.4. Installing Packages

DNF allows you to install both a single package and multiple packages, as well as a package group of your choice.

Installing Individual Packages

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

```
dnf install package_name
```


You can also install multiple packages simultaneously by appending their names as arguments:

```
dnf install package_name package_name...
```

If you are installing packages on a *multilib* system, such as an AMD64 or Intel64 machine, you can specify the architecture of the package, as long as it is available in an enabled repository, by appending *.arch* to the package name. For example, to install the *sqlite2* package for *i586*, type:

```
~]# dnf install sqlite2.i586
```

You can use glob expressions to quickly install multiple similarly-named packages:

```
~]# dnf install audacious-plugins-*
```

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

```
~]# dnf install /usr/sbin/named
```

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



Finding which package owns a file

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

```
~]# dnf provides "**bin/named"
Using metadata from Thu Apr 16 13:41:45 2015 (4:23:50 hours old)
bind-32:9.10.2-1.fc22.x86_64 : The Berkeley Internet Name Domain (BIND) DNS (Domain Name
System) server
Repo : @System
```

dnf provides *"*/file_name"* will find all the packages that contain *file_name*.

Installing a Package Group

A package group is similar to a package: it is not useful by itself, but installing one pulls a group of dependent packages that serve a common purpose. A package group has a name and a *groupid* (GID). The **dnf group list -v** command lists the names of all package groups, and, next to each of them, their groupid in parentheses. The groupid is always the term in the last pair of parentheses, such as **kde-desktop-environment** in the following example:

```
~]# dnf -v group list kde*
cachedir: /var/cache/dnf/x86_64/22
Loaded plugins: builddep, config-manager, copr, playground, debuginfo-install, download,
generate_completion_cache, kickstart, needs-restarting, noroot, protected_packages, Query,
reposync, langpacks
initialized Langpacks plugin
```



```
DNF version: 0.6.5
repo: using cache for: fedora
not found deltainfo for: Fedora 22 - x86_64
not found updateinfo for: Fedora 22 - x86_64
repo: using cache for: updates-testing
repo: using cache for: updates
not found updateinfo for: Fedora 22 - x86_64 - Updates
Using metadata from Thu Apr 16 13:41:45 2015 (4:37:51 hours old)
Available environment groups:
  KDE Plasma Workspaces (kde-desktop-environment)
```

You can install a package group by passing its full group name (without the groupid part) to **group install**:

```
dnf group install group_name
```

Multi-word names must be quoted.

You can also install by groupid:

```
dnf group install groupid
```

You can even pass the groupid, or quoted name, to the **install** command if you prepend it with an **@**-symbol (which tells **dnf** that you want to perform a **group install**):

```
dnf install @group
```

For example, the following are alternative but equivalent ways of installing the **KDE Plasma Workspaces** group:

```
~]# dnf group install "KDE Plasma Workspaces"
~]# dnf group install kde-desktop-environment
~]# dnf install @kde-desktop-environment
```

6.2.5. Removing Packages

Similarly to package installation, DNF allows you to uninstall (remove in **RPM** and DNF terminology) both individual packages and a package group.

Removing Individual Packages

To uninstall a particular package, as well as any packages that depend on it, run the following command as root:

```
dnf remove package_name...
```

As when you install multiple packages, you can remove several at once by adding more package names to the command. For example, to remove *totem*, *rhythmbox*, and *sound-juicer*, type the following at a shell prompt:

```
~]# dnf remove totem rhythmbox sound-juicer
```

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

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



Removing a package when other packages depend on it

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

Removing a Package Group

You can remove a package group using syntax congruent with the **install** syntax:

```
dnf group remove group
```

```
dnf remove @group
```

The following are alternative but equivalent ways of removing the **KDE Plasma Workspaces** group:

```
~]# dnf group remove "KDE Plasma Workspaces"
~]# dnf group remove kde-desktop-environment
~]# dnf remove @kde-desktop-environment
```

6.2.6. Working with Transaction History

The **dnf history** command allows users to review information about a timeline of DNF transactions, the dates and times on when they occurred, the number of packages affected, whether transactions succeeded or were aborted, and if the RPM database was changed between transactions. Additionally, this command can be used to undo or redo certain transactions.

Listing Transactions

To display a list of all transactions, as root, either run **dnf history** with no additional arguments, or enter the following command:

```
dnf history list
```

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

```
dnf history list start_id..end_id
```

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


```
dnf history list glob_expression...
```

For example, the list of first five transactions may look as follows:

```
~]# dnf history list 1..4
Using metadata from Thu Apr 16 13:41:45 2015 (5:47:31 hours old)
ID      | Login user          | Date a | Action | Altere
-----|-----|-----|-----|-----
  4 | root <root>         | 2015-04-16 18:35 | Erase      | 1
  3 | root <root>         | 2015-04-16 18:34 | Install    | 1
  2 | root <root>         | 2015-04-16 17:53 | Install    | 1
  1 | System <unset>      | 2015-04-16 14:14 | Install    | 668 E
```

The **dnf history list** command produces tabular output with each row consisting of the following columns:

- **ID** — an integer value that identifies a particular transaction.
- **Login user** — the name of the user whose login session was used to initiate a transaction. This information is typically presented in the **Full Name <username>** form, however sometimes the command used to perform the transaction is displayed. For transactions that were not issued by a user (such as an automatic system update), **System <unset>** is used instead.
- **Date and time** — the date and time when a transaction was issued.
- **Action(s)** — a list of actions that were performed during a transaction as described in [Table 6.1, “Possible values of the Action\(s\) field”](#).
- **Altered** — the number of packages that were affected by a transaction, possibly followed by additional information.

Table 6.1. Possible values of the Action(s) field

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

Reverting and Repeating Transactions

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

```
dnf history undo id
```

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

```
dnf history redo id
```

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

Note that both **dnf history undo** and **dnf history redo** commands merely revert or repeat the steps that were performed during a transaction, and will fail if the required packages are not available. For example, if the transaction installed a new package, the **dnf history undo** command will uninstall it and also attempt to downgrade all updated packages to their previous version, but the command will fail if the required packages are not available.

6.3. Configuring DNF and DNF Repositories

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

This section shows you how to:

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

6.3.1. Setting [main] Options

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

A sample **/etc/dnf/dnf.conf** configuration file can look like this:

```
[main]
gpgcheck=1
installonly_limit=3
clean_requirements_on_remove=true
```

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

debuglevel=value

...where *value* is an integer between **0** and **10**. Setting a higher **debuglevel** value causes **dnf** to display more detailed debugging output. **debuglevel=0** disables debugging output, and **debuglevel=2** is the default.

exclude=package_name [more_package_names]

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

gpgcheck=*value*

...where *value* is one of:

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

1 — Enable GPG signature-checking on all packages in all repositories, including local package installation. **gpgcheck=1** is the default, and thus all packages' signatures are checked.

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

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

installonlypkgs=*space separated list of packages*

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

If you add the **installonlypkgs** directive to **/etc/dnf/dnf.conf**, you should ensure that you list *all* of the packages that should be install-only, including any of those listed under the **installonlypkgs** section of **dnf.conf(5)**. In particular, kernel packages should always be listed in **installonlypkgs** (as they are by default), and **installonly_limit** should always be set to a value greater than **2** so that a backup kernel is always available in case the default one fails to boot.

installonly_limit=*value*

...where *value* is an integer representing the maximum number of versions that can be installed simultaneously for any single package listed in the **installonlypkgs** directive.

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

keepcache=*value*

...where *value* is one of:

0 — Do not retain the cache of headers and packages after a successful installation. This is the default.

1 — Retain the cache after a successful installation.

For a complete list of available **[main]** options, refer to the **[MAIN] OPTIONS** section of the **dnf.conf(5)** manual page.

6.3.2. Setting [repository] Options

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

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


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

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

name=repository_name

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

parameter=repository_url

...where *parameter* is one of the following: **baseurl**, **metalink**, or **mirrorlist**;

...where *repository_url* is a URL to a directory containing a **repodata** directory of a repository, a metalink file, or a mirror list file.

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

Usually this URL is an HTTP link, such as:

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

Note that DNF always expands the `$releasever`, `$arch`, and `$basearch` variables in URLs. For more information about DNF variables, refer to [Section 6.3.3, “Using DNF Variables”](#).

To configure the default set of repositories, use the **enabled** option as follows:

enabled=value

...where *value* is one of:

- 0** — Do not include this repository as a package source when performing updates and installs.
- 1** — Include this repository as a package source.

Turning repositories on and off can also be performed by passing either the **--set-enabled repo_name** or **--set-disabled repo_name** option to the **dnf** command, or through the **Add/Remove Software** window of the **PackageKit** utility.

Many more **[repository]** options exist. For a complete list, refer to the **[repository] OPTIONS** section of the **dnf.conf(5)** manual page.

6.3.3. Using DNF Variables

Variables can be used only in the appropriate sections of the DNF configuration files, namely the `/etc/dnf/dnf.conf` file and all `.repo` files in the `/etc/yum.repos.d/` directory. Repository variables include:

\$releasever

Refers to the release version of operating system which DNF derives from information available in RPMDB.

\$arch

Refers to the system's CPU architecture. Valid values for **\$arch** include: **i586**, **i686** and **x86_64**.

\$basearch

Refers to the base architecture of the system. For example, i686 and i586 machines both have a base architecture of **i386**, and AMD64 and Intel64 machines have a base architecture of **x86_64**.

6.4. Viewing the Current Configuration

To list all configuration options and their corresponding values, and the repositories, execute the **dnf config-manager** command with the **--dump** option:

```
~]$ dnf config-manager --dump
===== main =====
[main]
alwaysprompt = True
assumeno = False
assumeyes = False
bandwidth = 0
best = False
bugtracker_url = https://bugzilla.redhat.com/enter_bug.cgi?product=Fedora&component=dnf
cachedir = /var/cache/dnf/x86_64/22
[output truncated]
```

6.5. Adding, Enabling, and Disabling a DNF Repository

[Section 6.3.2, “Setting \[repository\] Options”](#) describes various options you can use to define a DNF repository. This section explains how to add, enable, and disable a repository by using the **dnf config-manager** command.

Adding a DNF Repository

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

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

```
dnf config-manager --add-repo repository_url
```

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

Example 6.8. Adding example.repo

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

```
~]# dnf config-manager --add-repo http://www.example.com/example.repo
```



```
adding repo from: http://www.example.com/example.repo
```

Enabling a DNF Repository

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

```
dnf config-manager --set-enabled repository...
```

...where *repository* is the unique repository ID. To display the current configuration, add the **--dump** option.

Disabling a DNF Repository

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

```
dnf config-manager --set-disabled repository...
```

...where *repository* is the unique repository ID. To display the current configuration, add the **--dump** option.

6.6. Additional Resources

Installed Documentation

- **dnf(8)** — The DNF command reference manual page.
- **dnf.conf(8)** — DNF Configuration Reference manual page.

Online Documentation

<http://dnf.readthedocs.org/en/latest/index.html>

The DNF wiki contains more documentation.

Part III. Infrastructure Services

This part provides information on how to configure services and daemons, configure authentication, and enable remote logins.

Services and Daemons

Maintaining security on your system is extremely important, and one approach for this task is to manage access to system services carefully. Your system may need to provide open access to particular services (for example, `httpd` if you are running a web server). However, if you do not need to provide a service, you should turn it off to minimize your exposure to possible bug exploits.

This chapter covers the configuration of the services to be run when a system is started, and provides information on how to start, stop, and restart the services on the command line using the **systemctl** utility.



Keep the system secure

When you allow access for new services, always remember that both the firewall and **SELinux** need to be configured as well. One of the most common mistakes committed when configuring a new service is neglecting to implement the necessary firewall configuration and SELinux policies to allow access for it. For more information, refer to the *Fedora 26 Security Guide*.

7.1. Configuring Services

To allow you to configure which services are started at boot time, Fedora is shipped with the **systemctl** command line tool.



Do not use the `ntsysv` and `chkconfig` utilities

Although it is still possible to use the **ntsysv** and **chkconfig** utilities to manage services that have init scripts installed in the `/etc/rc.d/init.d/` directory, it is advised that you use the **systemctl** utility.



Enabling the `irqbalance` service

To ensure optimal performance on POWER architecture, it is recommended that the `irqbalance` service is enabled. In most cases, this service is installed and configured to run during the Fedora 26 installation. To verify that `irqbalance` is running, type the following at a shell prompt:

```
systemctl status irqbalance.service
```

7.1.1. Enabling the Service

To configure a service to be automatically started at boot time, use the **systemctl** command in the following form:


```
systemctl enable service_name.service
```

The service will be started the next time you boot the system. For information on how to start the service immediately, refer to [Section 7.2.2, “Running the Service”](#).

Example 7.1. Enabling the httpd service

Imagine you want to run the Apache HTTP Server on your system. Provided that you have the *httpd* package installed, you can enable the *httpd* service by typing the following at a shell prompt as root:

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

7.1.2. Disabling the Service

To disable starting a service at boot time, use the **systemctl** command in the following form:

```
systemctl disable service_name.service
```

The next time you boot the system, the service will *not* be started. For information on how to stop the service immediately, refer to [Section 7.2.3, “Stopping the Service”](#).

Example 7.2. Disabling the telnet service

In order to secure the system, users are advised to disable insecure connection protocols such as Telnet. You can make sure that the *telnet* service is disabled by running the following command as root:

```
~]# systemctl disable telnet.service
```

7.2. Running Services

The **systemctl** utility also allows you to determine the status of a particular service, as well as to start, stop, or restart a service.

Do not use the service utility

Although it is still possible to use the **service** utility to manage services that have init scripts installed in the */etc/rc.d/init.d/* directory, it is advised that you use the **systemctl** utility.

7.2.1. Checking the Service Status

To determine the status of a particular service, use the **systemctl** command in the following form:

```
systemctl status service_name.service
```

This command provides detailed information on the service's status. However, if you merely need to verify that a service is running, you can use the **systemctl** command in the following form instead:


```
systemctl is-active service_name.service
```

Example 7.3. Checking the status of the httpd service

[Example 7.1, “Enabling the httpd service”](#) illustrated how to enable starting the httpd service at boot time. Imagine that the system has been restarted and you need to verify that the service is really running. You can do so by typing the following at a shell prompt:

```
~]$ systemctl is-active httpd.service
active
```

You can also display detailed information about the service by running the following command:

```
~]$ systemctl status httpd.service
httpd.service - LSB: start and stop Apache HTTP Server
   Loaded: loaded (/etc/rc.d/init.d/httpd)
   Active: active (running) since Mon, 23 May 2011 21:38:57 +0200; 27s ago
   Process: 2997 ExecStart=/etc/rc.d/init.d/httpd start (code=exited, status=0/
SUCCESS)
   Main PID: 3002 (httpd)
   CGroup: name=systemd:/system/httpd.service
           └─ 3002 /usr/sbin/httpd
           └─ 3004 /usr/sbin/httpd
           └─ 3005 /usr/sbin/httpd
           └─ 3006 /usr/sbin/httpd
           └─ 3007 /usr/sbin/httpd
           └─ 3008 /usr/sbin/httpd
           └─ 3009 /usr/sbin/httpd
           └─ 3010 /usr/sbin/httpd
           └─ 3011 /usr/sbin/httpd
```

To display a list of all active system services, use the following command:

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

This command provides a tabular output with each line consisting of the following columns:

- **UNIT** — A systemd unit name. In this case, a service name.
- **LOAD** — Information whether the systemd unit was properly loaded.
- **ACTIVE** — A high-level unit activation state.
- **SUB** — A low-level unit activation state.
- **JOB** — A pending job for the unit.
- **DESCRIPTION** — A brief description of the unit.

Example 7.4. Listing all active services

You can list all active services by using the following command:

```
~]$ systemctl list-units --type=service
UNIT                                LOAD    ACTIVE SUB    JOB DESCRIPTION
abrt-cpp.service                    loaded active exited    LSB: Installs coredump handler which
  saves segfault data
abrt-oops.service                   loaded active running    LSB: Watches system log for oops
  messages, creates ABRT dump directories for each oops
```



```
abrt.service      loaded active running  ABRT Automated Bug Reporting Tool
accounts-daemon.service loaded active running  Accounts Service
atd.service       loaded active running  Job spooling tools
[output truncated]
```

In the example above, the `abrt` service is loaded, active, and running, and it does not have any pending jobs.

7.2.2. Running the Service

To run a service, use the **systemctl** command in the following form:

```
systemctl start service_name.service
```

This will start the service in the current session. To configure the service to be started at boot time, refer to [Section 7.1.1, “Enabling the Service”](#).

Example 7.5. Running the `httpd` service

[Example 7.1, “Enabling the `httpd` service”](#) illustrated how to run the `httpd` service at boot time. You can start the service immediately by typing the following at a shell prompt as root:

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

7.2.3. Stopping the Service

To stop a service, use the **systemctl** command in the following form:

```
systemctl stop service_name.service
```

This will stop the service in the current session. To disable starting the service at boot time, refer to [Section 7.1.1, “Enabling the Service”](#).

Example 7.6. Stopping the `telnet` service

[Example 7.2, “Disabling the `telnet` service”](#) illustrated how to disable starting the `telnet` service at boot time. You can stop the service immediately by running the following command as root:

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

7.2.4. Restarting the Service

To restart a service, use the **systemctl** command in the following form:

```
systemctl restart service_name.service
```

Example 7.7. Restarting the `sshd` service

For any changes in the `/etc/ssh/sshd_config` configuration file to take effect, it is required that you restart the `sshd` service. You can do so by typing the following at a shell prompt as root:

```
~]# systemctl restart sshd.service
```


7.3. Additional Resources

7.3.1. Installed Documentation

- **systemctl(1)** — The manual page for the **systemctl** utility.

7.3.2. Related Books

Fedora 26 Security Guide

A guide to securing Fedora. It contains valuable information on how to set up the firewall, as well as the configuration of **SELinux**.

OpenSSH

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

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

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

8.1. The SSH Protocol

8.1.1. Why Use SSH?

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

Interception of communication between two systems

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

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

Impersonation of a particular host

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

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

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

8.1.2. Main Features

The SSH protocol provides the following safeguards:

No one can pose as the intended server

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

No one can capture the authentication information

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

No one can intercept the communication

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

Additionally, it also offers the following options:

It provides secure means to use graphical applications over a network

Using a technique called *X11 forwarding*, the client can forward *X11 (X Window System)* applications from the server. Note that if you set the **ForwardX11Trusted** option to **yes** or you use SSH with the **-Y** option, you bypass the X11 SECURITY extension controls, which can result in a security threat.

It provides a way to secure otherwise insecure protocols

The SSH protocol encrypts everything it sends and receives. Using a technique called *port forwarding*, an SSH server can become a conduit to securing otherwise insecure protocols, like POP, and increasing overall system and data security.

It can be used to create a secure channel

The OpenSSH server and client can be configured to create a tunnel similar to a virtual private network for traffic between server and client machines.

It supports the Kerberos authentication

OpenSSH servers and clients can be configured to authenticate using the GSSAPI (Generic Security Services Application Program Interface) implementation of the Kerberos network authentication protocol.

8.1.3. Protocol Versions

Two varieties of SSH currently exist: version 1 and version 2. The OpenSSH suite under Fedora uses SSH version 2, which has an enhanced key exchange algorithm not vulnerable to the known exploit in version 1. However, for compatibility reasons, the OpenSSH suite does support version 1 connections as well, although version 1 is disabled by default and needs to be enabled in the configuration files.



Avoid using SSH version 1

To ensure maximum security for your connection, it is recommended that only SSH version 2-compatible servers and clients are used whenever possible.

8.1.4. Event Sequence of an SSH Connection

The following series of events help protect the integrity of SSH communication between two hosts.

1. A cryptographic handshake is made so that the client can verify that it is communicating with the correct server.
2. The transport layer of the connection between the client and remote host is encrypted using a symmetric cipher.
3. The client authenticates itself to the server.
4. The client interacts with the remote host over the encrypted connection.

8.1.4.1. Transport Layer

The primary role of the transport layer is to facilitate safe and secure communication between the two hosts at the time of authentication and during subsequent communication. The transport layer accomplishes this by handling the encryption and decryption of data, and by providing integrity protection of data packets as they are sent and received. The transport layer also provides compression, speeding the transfer of information.

Once an SSH client contacts a server, key information is exchanged so that the two systems can correctly construct the transport layer. The following steps occur during this exchange:

- Keys are exchanged
- The public key encryption algorithm is determined
- The symmetric encryption algorithm is determined
- The message authentication algorithm is determined
- The hash algorithm is determined

During the key exchange, the server identifies itself to the client with a unique *host key*. If the client has never communicated with this particular server before, the server's host key is unknown to the client and it does not connect. OpenSSH notifies the user that the authenticity of the host cannot be established and prompts the user to accept or reject it. The user is expected to independently verify the new host key before accepting it. In subsequent connections, the server's host key is checked against the saved version on the client, providing confidence that the client is indeed communicating with the intended server. If, in the future, the host key no longer matches, the user must remove the client's saved version before a connection can occur.



Always verify the integrity of a new SSH server

It is possible for an attacker to masquerade as an SSH server during the initial contact since the local system does not know the difference between the intended server and a false one set up by an attacker. To help prevent this, verify the integrity of a new SSH server by contacting the server administrator before connecting for the first time or in the event of a host key mismatch.

SSH is designed to work with almost any kind of public key algorithm or encoding format. After an initial key exchange creates a hash value used for exchanges and a shared secret value, the two systems immediately begin calculating new keys and algorithms to protect authentication and future data sent over the connection.

After a certain amount of data has been transmitted using a given key and algorithm (the exact amount depends on the SSH implementation), another key exchange occurs, generating another set of hash values and a new shared secret value. Even if an attacker is able to determine the hash and shared secret value, this information is only useful for a limited period of time.

8.1.4.2. Authentication

Once the transport layer has constructed a secure tunnel to pass information between the two systems, the server tells the client the different authentication methods supported, such as using a private key-encoded signature or typing a password. The client then tries to authenticate itself to the server using one of these supported methods.

SSH servers and clients can be configured to allow different types of authentication, which gives each side the optimal amount of control. The server can decide which encryption methods it supports based on its security model, and the client can choose the order of authentication methods to attempt from the available options.

8.1.4.3. Channels

After a successful authentication over the SSH transport layer, multiple channels are opened via a technique called *multiplexing*¹. Each of these channels handles communication for different terminal sessions and for forwarded X11 sessions.

Both clients and servers can create a new channel. Each channel is then assigned a different number on each end of the connection. When the client attempts to open a new channel, the client sends the channel number along with the request. This information is stored by the server and is used to direct communication to that channel. This is done so that different types of sessions do not affect one another and so that when a given session ends, its channel can be closed without disrupting the primary SSH connection.

Channels also support *flow-control*, which allows them to send and receive data in an orderly fashion. In this way, data is not sent over the channel until the client receives a message that the channel is open.

The client and server negotiate the characteristics of each channel automatically, depending on the type of service the client requests and the way the user is connected to the network. This allows great flexibility in handling different types of remote connections without having to change the basic infrastructure of the protocol.

8.2. Configuring OpenSSH

In order to perform tasks described in this section, you must have superuser privileges. To obtain them, log in as root by typing:

¹ A multiplexed connection consists of several signals being sent over a shared, common medium. With SSH, different channels are sent over a common secure connection.


```
su -
```

8.2.1. Configuration Files

There are two different sets of configuration files: those for client programs (that is, **ssh**, **scp**, and **sftp**), and those for the server (the **sshd** daemon).

System-wide SSH configuration information is stored in the `/etc/ssh/` directory as described in [Table 8.1, “System-wide configuration files”](#). User-specific SSH configuration information is stored in `~/.ssh/` within the user's home directory as described in [Table 8.2, “User-specific configuration files”](#).

Table 8.1. System-wide configuration files

File	Description
<code>/etc/ssh/moduli</code>	Contains Diffie-Hellman groups used for the Diffie-Hellman key exchange which is critical for constructing a secure transport layer. When keys are exchanged at the beginning of an SSH session, a shared, secret value is created which cannot be determined by either party alone. This value is then used to provide host authentication.
<code>/etc/ssh/ssh_config</code>	The default SSH client configuration file. Note that it is overridden by <code>~/.ssh/config</code> if it exists.
<code>/etc/ssh/sshd_config</code>	The configuration file for the sshd daemon.
<code>/etc/ssh/ssh_host_ecdsa_key</code>	The ECDSA private key used by the sshd daemon.
<code>/etc/ssh/ssh_host_ecdsa_key.pub</code>	The ECDSA public key used by the sshd daemon.
<code>/etc/ssh/ssh_host_key</code>	The RSA private key used by the sshd daemon for version 1 of the SSH protocol.
<code>/etc/ssh/ssh_host_key.pub</code>	The RSA public key used by the sshd daemon for version 1 of the SSH protocol.
<code>/etc/ssh/ssh_host_rsa_key</code>	The RSA private key used by the sshd daemon for version 2 of the SSH protocol.
<code>/etc/ssh/ssh_host_rsa_key.pub</code>	The RSA public key used by the sshd daemon for version 2 of the SSH protocol.
<code>/etc/pam.d/sshd</code>	The PAM configuration file for the sshd daemon.
<code>/etc/sysconfig/sshd</code>	Configuration file for the sshd service.

Table 8.2. User-specific configuration files

File	Description
<code>~/.ssh/authorized_keys</code>	Holds a list of authorized public keys for servers. When the client connects to a server, the server authenticates the client by checking its signed public key stored within this file.
<code>~/.ssh/id_ecdsa</code>	Contains the ECDSA private key of the user.
<code>~/.ssh/id_ecdsa.pub</code>	The ECDSA public key of the user.
<code>~/.ssh/id_rsa</code>	The RSA private key used by ssh for version 2 of the SSH protocol.

File	Description
<code>~/.ssh/id_rsa.pub</code>	The RSA public key used by ssh for version 2 of the SSH protocol.
<code>~/.ssh/identity</code>	The RSA private key used by ssh for version 1 of the SSH protocol.
<code>~/.ssh/identity.pub</code>	The RSA public key used by ssh for version 1 of the SSH protocol.
<code>~/.ssh/known_hosts</code>	Contains host keys of SSH servers accessed by the user. This file is very important for ensuring that the SSH client is connecting to the correct SSH server.

For information concerning various directives that can be used in the SSH configuration files, see the **ssh_config(5)** and **sshd_config(5)** manual pages.

8.2.2. Starting an OpenSSH Server

Make sure you have relevant packages installed

To run an OpenSSH server, you must have the *openssh-server* package installed. See [Section 6.2.4, “Installing Packages”](#) for more information on how to install new packages in Fedora 26.

To start the **sshd** daemon in the current session, type the following at a shell prompt as root:

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

To stop the running **sshd** daemon in the current session, use the following command as root:

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

If you want the daemon to start automatically at the boot time, type as root:

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

See [Chapter 7, Services and Daemons](#) for more information on how to configure services in Fedora.

Note that if you reinstall the system, a new set of identification keys will be created. As a result, clients who had connected to the system with any of the OpenSSH tools before the reinstall will see the following message:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```



```
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!    @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
```

To prevent this, you can backup the relevant files from the `/etc/ssh/` directory (see [Table 8.1, “System-wide configuration files”](#) for a complete list), and restore them whenever you reinstall the system.

8.2.3. Requiring SSH for Remote Connections

For SSH to be truly effective, using insecure connection protocols should be prohibited. Otherwise, a user's password may be protected using SSH for one session, only to be captured later while logging in using Telnet. Some services to disable include **telnet**, **rsh**, **rlogin**, and **vsftpd**.

These services are not installed by default in Fedora. If required, to make sure these services are not running, type the following commands at a shell prompt:

```
systemctl stop telnet.service
systemctl stop rsh.service
systemctl stop rlogin.service
systemctl stop vsftpd.service
```

To disable running these services at startup, type:

```
systemctl disable telnet.service
systemctl disable rsh.service
systemctl disable rlogin.service
systemctl disable vsftpd.service
```

See [Chapter 7, Services and Daemons](#) for more information on how to configure services in Fedora.

8.2.4. Using Key-based Authentication

To improve the system security even further, generate SSH key pairs and then enforce key-based authentication by disabling password authentication. To do so, open the `/etc/ssh/sshd_config` configuration file in a text editor such as **vi** or **nano**, and change the **PasswordAuthentication** option as follows:

```
PasswordAuthentication no
```

If you are working on a system other than a new default installation, check that **PubkeyAuthentication no** has **not** been set. If connected remotely, not using console or out-of-band access, testing the key-based log in process before disabling password authentication is advised.

To be able to use **ssh**, **scp**, or **sftp** to connect to the server from a client machine, generate an authorization key pair by following the steps below. Note that keys must be generated for each user separately.

Fedora 26 uses SSH Protocol 2 and RSA keys by default (see [Section 8.1.3, “Protocol Versions”](#) for more information).



Do not generate key pairs as root

If you complete the steps as root, only root will be able to use the keys.



Backup your ~/.ssh/ directory

If you reinstall your system and want to keep previously generated key pairs, backup the ~/.ssh/ directory. After reinstalling, copy it back to your home directory. This process can be done for all users on your system, including root.

8.2.4.1. Generating Key Pairs

To generate an RSA key pair for version 2 of the SSH protocol, follow these steps:

1. Generate an RSA key pair by typing the following at a shell prompt:

```
~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/USER/.ssh/id_rsa):
```

2. Press **Enter** to confirm the default location, ~/.ssh/id_rsa, for the newly created key.
3. Enter a passphrase, and confirm it by entering it again when prompted to do so. For security reasons, avoid using the same password as you use to log in to your account.

After this, you will be presented with a message similar to this:

```
Your identification has been saved in /home/USER/.ssh/id_rsa.
Your public key has been saved in /home/USER/.ssh/id_rsa.pub.
The key fingerprint is:
e7:97:c7:e2:0e:f9:0e:fc:c4:d7:cb:e5:31:11:92:14 USER@penguin.example.com
The key's randomart image is:
+--[ RSA 2048 ]-----+
|           E.        |
|          . .       |
|           o .       |
|            . .      |
|         S . .       |
|        + o o . .    |
|       * * +oo       |
|        0 +. . =     |
|       o*  o.        |
+-----+

```

4. By default, the permissions of the ~/.ssh/ directory are set to **rwX-----** or **700** expressed in octal notation. This is to ensure that only the *USER* can view the contents. If required, this can be confirmed with the following command:


```
~]$ ls -ld ~/.ssh
drwx----- . 2 USER USER 54 Nov 25 16:56 /home/USER/.ssh/
```

- To copy the public key to a remote machine, issue a command in the following format:

```
ssh-copy-id user@hostname
```

This will copy the most recently modified `~/.ssh/id*.pub` public key if it is not yet installed. Alternatively, specify the public key's file name as follows:

```
ssh-copy-id -i ~/.ssh/id_rsa.pub user@hostname
```

This will copy the content of `~/.ssh/id_rsa.pub` into the `~/.ssh/authorized_keys` file on the machine to which you want to connect. If the file already exists, the keys are appended to its end.

To generate an ECDSA key pair for version 2 of the SSH protocol, follow these steps:

- Generate an ECDSA key pair by typing the following at a shell prompt:

```
~]$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/USER/.ssh/id_ecdsa):
```

- Press **Enter** to confirm the default location, `~/.ssh/id_ecdsa`, for the newly created key.
- Enter a passphrase, and confirm it by entering it again when prompted to do so. For security reasons, avoid using the same password as you use to log in to your account.

After this, you will be presented with a message similar to this:

```
Your identification has been saved in /home/USER/.ssh/id_ecdsa.
Your public key has been saved in /home/USER/.ssh/id_ecdsa.pub.
The key fingerprint is:
fd:1d:ca:10:52:96:21:43:7e:bd:4c:fc:5b:35:6b:63 USER@penguin.example.com
The key's randomart image is:
+--[ECDSA 256]--+
|    .+ +o    |
|    . =.o    |
|    o o + .. |
|    + + o +  |
|    S o o oE. |
|    + oo+.   |
|    + o      |
|             |
+-----+
```

- By default, the permissions of the `~/.ssh/` directory are set to `rwX-----` or `700` expressed in octal notation. This is to ensure that only the *USER* can view the contents. If required, this can be confirmed with the following command:

```
~]$ ls -ld ~/.ssh
```



```
~]$ ls -ld ~/.ssh/  
drwx----- . 2 USER USER 54 Nov 25 16:56 /home/USER/.ssh/
```

5. To copy the public key to a remote machine, issue a command in the following format:

```
ssh-copy-id USER@hostname
```

This will copy the most recently modified `~/.ssh/id*.pub` public key if it is not yet installed. Alternatively, specify the public key's file name as follows:

```
ssh-copy-id -i ~/.ssh/id_ecdsa.pub USER@hostname
```

This will copy the content of `~/.ssh/id_ecdsa.pub` into the `~/.ssh/authorized_keys` on the machine to which you want to connect. If the file already exists, the keys are appended to its end.

See [Section 8.2.4.2, “Configuring ssh-agent”](#) for information on how to set up your system to remember the passphrase.



Never share your private key

The private key is for your personal use only, and it is important that you never give it to anyone.

8.2.4.2. Configuring ssh-agent

To store your passphrase so that you do not have to enter it each time you initiate a connection with a remote machine, you can use the **ssh-agent** authentication agent. To save your passphrase for a certain shell prompt, use the following command:

```
~]$ ssh-add  
Enter passphrase for /home/USER/.ssh/id_rsa:
```

Note that when you log out, your passphrase will be forgotten. You must execute the command each time you log in to a virtual console or a terminal window.

8.3. Using OpenSSH Certificate Authentication

8.3.1. Introduction to SSH Certificates

Using public key cryptography for authentication requires copying the public key from every client to every server that the client intends to log into. This system does not scale well and can be an administrative burden. Using a public key from a *certificate authority* (CA) to authenticate client certificates removes the need to copy keys between multiple systems. While the X.509 Public Key Infrastructure Certificate system provides a solution to this issue, there is a submission and validation process, with associated fees, to go through in order to get a certificate signed. As an alternative, OpenSSH supports the creation of simple certificates and associated CA infrastructure.

OpenSSH certificates contain a public key, identity information, and validity constraints. They are signed with a standard SSH public key using the **ssh-keygen** utility. The format of the certificate is described in `/usr/share/doc/openssh-version/PROTOCOL.certkeys`.

The **ssh-keygen** utility supports two types of certificates: user and host. User certificates authenticate users to servers, whereas host certificates authenticate server hosts to users. For certificates to be used for user or host authentication, **sshd** must be configured to trust the CA public key.

8.3.2. Support for SSH Certificates

Support for certificate authentication of users and hosts using the new OpenSSH certificate format was introduced in Fedora 13, in the *openssh-5.4p1-1.fc13* package. If required, to ensure the latest OpenSSH package is installed, enter the following command as root:

```
~]# dnf install openssh
Last metadata expiration check performed 0:58:01 ago on Sun Sep 6 16:07:22 2015.
Package openssh-7.1p1-1.fc23.x86_64 is already installed, skipping.
```

8.3.3. Creating SSH CA Certificate Signing Keys

Two types of certificates are required, host certificates and user certificates. It is considered better to have two separate keys for signing the two certificates, for example **ca_user_key** and **ca_host_key**, however it is possible to use just one CA key to sign both certificates. It is also easier to follow the procedures if separate keys are used, so the examples that follow will use separate keys.

The basic format of the command to sign user's public key to create a user certificate is as follows:

```
ssh-keygen -s ca_user_key -I certificate_ID id_rsa.pub
```

Where **-s** indicates the private key used to sign the certificate, **-I** indicates an identity string, the *certificate_ID*, which can be any alpha numeric value. It is stored as a zero terminated string in the certificate. The *certificate_ID* is logged whenever the certificate is used for identification and it is also used when revoking a certificate. Having a long value would make logs hard to read, therefore using the host name for host certificates and the user name for user certificates is a safe choice.

To sign a host's public key to create a host certificate, add the **-h** option:

```
ssh-keygen -s ca_host_key -I certificate_ID -h ssh_host_rsa_key.pub
```

Host keys are generated on the system by default, to list the keys, enter a command as follows:

```
~]# ls -l /etc/ssh/ssh_host*
-rw-----. 1 root root 668 Jul 9 2014 /etc/ssh/ssh_host_dsa_key
-rw-r--r--. 1 root root 590 Jul 9 2014 /etc/ssh/ssh_host_dsa_key.pub
-rw-----. 1 root root 963 Jul 9 2014 /etc/ssh/ssh_host_key
-rw-r--r--. 1 root root 627 Jul 9 2014 /etc/ssh/ssh_host_key.pub
-rw-----. 1 root root 1671 Jul 9 2014 /etc/ssh/ssh_host_rsa_key
-rw-r--r--. 1 root root 382 Jul 9 2014 /etc/ssh/ssh_host_rsa_key.pub
```




Important

It is recommended to create and store CA keys in a safe place just as with any other private key. In these examples the root user will be used. In a real production environment using an offline computer with an administrative user account is recommended. For guidance on key lengths see [NIST Special Publication 800-131A](http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf)².

Procedure 8.1. Generating SSH CA Certificate Signing Keys

1. On the server designated to be the CA, generate two keys for use in signing certificates. These are the keys that all other hosts need to trust. Choose suitable names, for example **ca_user_key** and **ca_host_key**. To generate the user certificate signing key, enter the following command as root:

```
~]# ssh-keygen -t rsa -f ~/.ssh/ca_user_key
Generating public/private rsa key pair.
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/ca_user_key.
Your public key has been saved in /root/.ssh/ca_user_key.pub.
The key fingerprint is:
11:14:2f:32:fd:5d:f5:e4:7a:5a:d6:b6:a0:62:c9:1f root@host_name.example.com
The key's randomart image is:
+--[ RSA 2048 ]-----+
|      .+.      0|
|      . o      +.|
|     o + .      o|
|      o + . . . |
|      S . . . *|
|      . . . . *|
|      = E . . |
|      . o .    |
|      .        |
+-----+
```

Generate a host certificate signing key, **ca_host_key**, as follows:

```
~]# ssh-keygen -t rsa -f ~/.ssh/ca_host_key
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/ca_host_key.
Your public key has been saved in /root/.ssh/ca_host_key.pub.
The key fingerprint is:
e4:d5:d1:4f:6b:fd:a2:e3:4e:5a:73:52:91:0b:b7:7a root@host_name.example.com
The key's randomart image is:
+--[ RSA 2048 ]-----+
|      ..      |
|      . . . . |
|     . . o +oo|
|     o .   o *o|
|      S     = .|
|      o . .   |
+-----+
```

² <http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf>


```
|          *.E. |
|          +0=  |
|          .00.  |
|-----+-----|
```

If required, confirm the permissions are correct:

```
~]# ls -la ~/.ssh
total 40
drwxrwxrwx. 2 root root 4096 May 22 13:18 .
dr-xr-x---. 3 root root 4096 May  8 08:34 ..
-rw-----. 1 root root 1743 May 22 13:15 ca_host_key
-rw-r--r--. 1 root root  420 May 22 13:15 ca_host_key.pub
-rw-----. 1 root root 1743 May 22 13:14 ca_user_key
-rw-r--r--. 1 root root  420 May 22 13:14 ca_user_key.pub
-rw-r--r--. 1 root root  854 May  8 05:55 known_hosts
-r-----. 1 root root 1671 May  6 17:13 ssh_host_rsa
-rw-r--r--. 1 root root 1370 May  7 14:30 ssh_host_rsa-cert.pub
-rw-----. 1 root root  420 May  6 17:13 ssh_host_rsa.pub
```

2. Create the CA server's own host certificate by signing the server's host public key together with an identification string such as the host name, the CA server's *fully qualified domain name* (FQDN) but without the trailing ., and a validity period. The command takes the following form:

```
ssh-keygen -s ~/.ssh/ca_host_key -I certificate_ID -h -n host_name.example.com -V -
start:+end /etc/ssh/ssh_host_rsa.pub
```

The **-n** option restricts this certificate to a specific host within the domain. The **-V** option is for adding a validity period; this is highly recommend. Where the validity period is intended to be one year, fifty two weeks, consider the need for time to change the certificates and any holiday periods around the time of certificate expiry.

For example:

```
~]# ssh-keygen -s ~/.ssh/ca_host_key -I host_name -h -n host_name.example.com -V -1w:
+54w5d /etc/ssh/ssh_host_rsa.pub
Enter passphrase:
Signed host key /root/.ssh/ssh_host_rsa-cert.pub: id "host_name" serial 0 for
host_name.example.com valid from 2015-05-15T13:52:29 to 2016-06-08T13:52:29
```

8.3.4. Distributing and Trusting SSH CA Public Keys

Hosts that are to allow certificate authenticated log in from users must be configured to trust the CA's public key that was used to sign the user certificates, in order to authenticate user's certificates. In this example that is the **ca_user_key.pub**.

Publish the **ca_user_key.pub** key and download it to all hosts that are required to allow remote users to log in. Alternately, copy the CA user public key to all the hosts. In a production environment, consider copying the public key to an administrator account first. The secure copy command can be used to copy the public key to remote hosts. The command has the following format:

```
scp ~/.ssh/ca_user_key.pub root@host_name.example.com:/etc/ssh/
```

Where *host_name* is the host name of a server the is required to authenticate user's certificates presented during the login process. Ensure you copy the public key not the private key. For example, as root:

```
~]# scp ~/.ssh/ca_user_key.pub root@host_name.example.com:/etc/ssh/
```



```
The authenticity of host 'host_name.example.com (10.34.74.56)' can't be established.  
RSA key fingerprint is fc:23:ad:ae:10:6f:d1:a1:67:ee:b1:d5:37:d4:b0:2f.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added 'host_name.example.com,10.34.74.56' (RSA) to the list of known  
hosts.  
root@host_name.example.com's password:  
ca_user_key.pub                               100% 420      0.4KB/s   00:00
```

For remote user authentication, CA keys can be marked as trusted per-user in the `~/.ssh/authorized_keys` file using the **cert-authority** directive or for global use by means of the **TrustedUserCAKeys** directive in the `/etc/ssh/sshd_config` file. For remote host authentication, CA keys can be marked as trusted globally in the `/etc/ssh/known_hosts` file or per-user in the `~/.ssh/ssh_known_hosts` file.

Procedure 8.2. Trusting the User Signing Key

- For user certificates which have one or more principles listed, and where the setting is to have global effect, edit the `/etc/ssh/sshd_config` file as follows:

```
TrustedUserCAKeys /etc/ssh/ca_user_key.pub
```

Restart `sshd` to make the changes take effect:

```
~]# service sshd restart
```

To avoid being presented with the warning about an unknown host, a user's system must trust the CA's public key that was used to sign the host certificates. In this example that is **ca_host_key.pub**.

Procedure 8.3. Trusting the Host Signing Key

- Extract the contents of the public key used to sign the host certificate. For example, on the CA:

```
cat ~/.ssh/ca_host_key.pub  
ssh-rsa AAAAB5Wm.== root@ca-server.example.com
```

- To configure client systems to trust servers' signed host certificates, add the contents of the **ca_host_key.pub** into the global **known_hosts** file. This will automatically check a server's host advertised certificate against the CA public key for all users every time a new machine is connected to in the domain ***.example.com**. Login as root and configure the `/etc/ssh/ssh_known_hosts` file, as follows:

```
~]# vi /etc/ssh/ssh_known_hosts  
# A CA key, accepted for any host in *.example.com  
@cert-authority *.example.com ssh-rsa AAAAB5Wm.
```

Where **ssh-rsa AAAAB5Wm.** is the contents of **ca_host_key.pub**. The above configures the system to trust the CA servers host public key. This enables global authentication of the certificates presented by hosts to remote users.

8.3.5. Creating SSH Certificates

A certificate is a signed public key. The user's and host's public keys must be copied to the CA server for signing by the CA server's private key.



Important

Copying many keys to the CA to be signed can create confusion if they are not uniquely named. If the default name is always used then the latest key to be copied will overwrite the previously copied key, which may be an acceptable method for one administrator. In the example below the default name is used. In a production environment, consider using easily recognizable names. It is recommended to have a designated directory on the CA server owned by an administrative user for the keys to be copied into. Copying these keys to the root user's `/etc/ssh/` directory is not recommended. In the examples below an account named `admin` with a directory named `keys/` will be used.

Create an administrator account, in this example `admin`, and a directory to receive the user's keys. For example:

```
~]$ mkdir keys
```

Set the permissions to allow keys to be copied in:

```
~]$ chmod o+w keys
ls -la keys
total 8
drwxrwxrwx. 2 admin admin 4096 May 22 16:17 .
drwx-----. 3 admin admin 4096 May 22 16:17 ..
```

8.3.5.1. Creating SSH Certificates to Authenticate Hosts

The command to sign a host certificate has the following format:

```
ssh-keygen -s ca_host_key -I host_name -h ssh_host_rsa_key.pub
```

The host certificate will be named `ssh_host_rsa_key-cert.pub`.

Procedure 8.4. Generating a Host Certificate

To authenticate a host to a user, a public key must be generated on the host, passed to the CA server, signed by the CA, and then passed back to be stored on the host to present to a user attempting to log into the host.

1. Host keys are generated automatically on the system. To list them enter the following command:

```
~]# ls -l /etc/ssh/ssh_host*
-rw-----. 1 root root 668 May 6 14:38 /etc/ssh/ssh_host_dsa_key
-rw-r--r--. 1 root root 590 May 6 14:38 /etc/ssh/ssh_host_dsa_key.pub
-rw-----. 1 root root 963 May 6 14:38 /etc/ssh/ssh_host_key
-rw-r--r--. 1 root root 627 May 6 14:38 /etc/ssh/ssh_host_key.pub
-rw-----. 1 root root 1679 May 6 14:38 /etc/ssh/ssh_host_rsa_key
-rw-r--r--. 1 root root 382 May 6 14:38 /etc/ssh/ssh_host_rsa_key.pub
```

2. Copy the chosen public key to the server designated as the CA. For example, from the host:

```
~]# scp /etc/ssh/ssh_host_rsa_key.pub admin@ca-server.example.com:~/keys/
ssh_host_rsa_key.pub
```



```
The authenticity of host 'ca-server.example.com (10.34.74.58)' can't be established.  
RSA key fingerprint is b0:e5:ea:b8:75:e2:f0:b1:fe:5b:07:39:7f:58:64:d9.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added 'ca-server.example.com,10.34.74.58' (RSA) to the list of  
known hosts.  
admin@ca-server.example.com's password:  
ssh_host_rsa_key.pub                               100% 382      0.4KB/s   00:00
```

Alternately, from the CA:

```
~]$ scp root@host_name.example.com:/etc/ssh/ssh_host_rsa_key.pub ~/keys/  
ssh_host_rsa_key.pub
```

3. On the CA server, sign the host's public key. For example, as root:

```
~]# ssh-keygen -s ~/.ssh/ca_host_key -I host_name -h -n host_name.example.com -V -id:  
+54w /home/admin/keys/ssh_host_rsa_key.pub  
Enter passphrase:  
Signed host key /home/admin/keys/ssh_host_rsa_key-cert.pub: id "host_name" serial 0 for  
host_name.example.com valid from 2015-05-26T12:21:54 to 2016-06-08T12:21:54
```

Where *host_name* is the host name of the system requiring the certificate.

4. Copy the certificate to the host. For example, from the CA:

```
~]# scp /home/admin/keys/ssh_host_rsa_key-cert.pub root@host_name.example.com:/etc/ssh/  
root@host_name.example.com's password:  
ssh_host_rsa_key-cert.pub                               100% 1384     1.5KB/s   00:00
```

5. Configure the host to present the certificate to a user's system when a user initiates the login process. As root, edit the **/etc/ssh/sshd_config** file as follows:

```
HostCertificate /etc/ssh/ssh_host_rsa_key-cert.pub
```

6. Restart sshd to make the changes take effect:

```
~]# service sshd restart
```

7. On user's systems, remove keys belonging to hosts from the **~/.ssh/known_hosts** file if the user has previously logged into the host configured above. When a user logs into the host they should no longer be presented with the warning about the hosts authenticity.

To test the host certificate, on a client system, ensure the client has set up the global **/etc/ssh/known_hosts** file, as described in [Procedure 8.3, "Trusting the Host Signing Key"](#), and that the server's public key is not in the **~/.ssh/known_hosts** file. Then attempt to log into the server over SSH as a remote user. You should not see a warning about the authenticity of the host. If required, add the **-v** option to the SSH command to see logging information.

8.3.5.2. Creating SSH Certificates for Authenticating Users

To sign a user's certificate, use a command in the following format:

```
ssh-keygen -s ca_user_key -I user_name -n user_name -V -start:+end id_rsa.pub
```

The resulting certificate will be named **id_rsa-cert.pub**.

The default behavior of OpenSSH is that a user is allowed to log in as a remote user if one of the principals specified in the certificate matches the remote user's name. This can be adjusted in the following ways:

- Add more user's names to the certificate during the signing process using the **-n** option:

```
-n "name1[, name2, ...]"
```

- On the user's system, add the public key of the CA in the `~/.ssh/authorized_keys` file using the **cert-authority** directive and list the principals names as follows:

```
~]# vi ~/.ssh/authorized_keys
# A CA key, accepted for any host in *.example.com
@cert-authority principals="name1,name2" *.example.com ssh-rsa AAAAB5Wm.
```

- On the server, create an **AuthorizedPrincipalsFile** file, either per user or globally, and add the principles' names to the file for those users allowed to log in. Then in the `/etc/ssh/sshd_config` file, specify the file using the **AuthorizedPrincipalsFile** directive.

Procedure 8.5. Generating a User Certificate

To authenticate a user to a remote host, a public key must be generated by the user, passed to the CA server, signed by the CA, and then passed back to be stored by the user for use when logging in to a host.

1. On client systems, login as the user who requires the certificate. Check for available keys as follows:

```
~]$ ls -l ~/.ssh/
```

If no suitable public key exists, generate one and set the directory permissions if the directory is not the default directory. For example, enter the following command:

```
~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user1/.ssh/id_rsa):
Created directory '/home/user1/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user1/.ssh/id_rsa.
Your public key has been saved in /home/user1/.ssh/id_rsa.pub.
The key fingerprint is:
b1:f8:26:a7:46:87:c3:60:54:a3:6d:85:0d:60:fe:ce user1@host1.example.com
The key's randomart image is:
+--[ RSA 2048 ]-----+
|      oo++      |
|      o.o.o      |
|      .o o .      |
|      oo . o      |
|      . oo.S      |
|      o=..        |
|      .Eo+        |
|      . =         |
|      ..          |
+-----+

```

By default the directory permissions for a user's keys are **drwx-----**, or octal 0700. If required, confirm the permissions are correct:

```
~]$ ls -la ~/.ssh
```



```
total 16
drwx-----. 2 user1 user1 4096 May  7 12:37 .
drwx-----. 3 user1 user1 4096 May  7 12:37 ..
-rw-----. 1 user1 user1 1679 May  7 12:37 id_rsa
-rw-r--r--. 1 user1 user1  421 May  7 12:37 id_rsa.pub
```

See [Section 8.2.4, “Using Key-based Authentication”](#) for more examples of key generation and for instructions on setting the correct directory permissions.

2. The chosen public key must be copied to the server designated as the CA, in order to be signed. The secure copy command can be used to do this, the command has the following format:

```
scp ~/.ssh/id_protocol.pub admin@ca_server.example.com:~/keys/
```

Where *protocol* is the part of the file name indicating the protocol used to generate the key, for example **rsa**, *admin* is an account on the CA server, and */keys/* is a directory setup to receive the keys to be signed.

Copy the chosen public key to the server designated as the CA. For example:

```
~]$ scp ~/.ssh/id_rsa.pub admin@ca-server.example.com:~/keys/
admin@ca-server.example.com's password:
id_rsa.pub                                100% 421      0.4KB/s   00:00
```

If you have configured the client system to trust the host signing key as described in [Procedure 8.3, “Trusting the Host Signing Key”](#) then you should not see a warning about the authenticity of the remote host.

3. On the CA server, sign the user's public key. For example, as root:

```
~]# ssh-keygen -s ~/.ssh/ca_user_key -I user1 -n user1 -V -id:+54w /home/admin/keys/
id_rsa.pub
Enter passphrase:
Signed user key /home/admin/keys/id_rsa-cert.pub: id "user1" serial 0 for
host_name.example.com valid from 2015-05-21T16:43:17 to 2016-06-03T16:43:17
```

4. Copy the resulting certificate to the user's *~/ .ssh/* directory on their system. For example:

```
~]# scp /home/admin/keys/id_rsa-cert.pub user1@host_name.example.com:~/ .ssh/
user1@host_name.example.com's password:
id_rsa-cert.pub                        100% 1498     1.5KB/s   00:00
```

5. If using the standard file names and location then no further configuration is required as the SSH daemon will search for user certificates ending in **-cert.pub** and use them automatically if it finds them. Note that the default location and file names for for SSH version 2 keys are: *~/ .ssh/ id_dsa*, *~/ .ssh/ id_ecdsa* and *~/ .ssh/ id_rsa* as explained in the **ssh_config(5)** manual page. If you use these locations and naming conventions then there is no need for editing the configuration files to enable **sshd** to present the certificate. They will be used automatically when logging in to a remote system. In this is the case then skip to step 6.

If required to use a non-default directory or file naming convention, then as root, add the following line to the */etc/ssh/ssh_config* or *~/ .ssh/config* files:

```
IdentityFile ~/path/key_file
```


Note that this must be the private key name, do not had **.pub** or **-cert.pub**. Ensure the file permission are correct. For example:

```
~]$ ls -la ~/.ssh/config
-rw-rw-r--. 1 user1 user1 36 May 27 21:49 /home/user1/.ssh/config
chmod 700 ~/.ssh/config
~]$ ls -la ~/.ssh/config
-rwx-----. 1 user1 user1 36 May 27 21:49 /home/user1/.ssh/config
```

This will enable the user of this system to be authenticated by a user certificate when logging into a remote system configured to trust the CA user certificate signing key.

6. To test the user certificate, attempt to log into a server over SSH from the user's account. You should do this as the user listed as a principle in the certificate, if any are specified. You should not be prompted for a password. If required, add the **-v** option to the SSH command to see logging information.

8.3.6. Signing an SSH Certificate Using a PKCS#11 Token

It is possible to sign a host key using a CA key stored in a PKCS#11 token by providing the token library using the **-D** and identifying the CA key by providing its public half as an argument to the **-s** option:

```
ssh-keygen -s ca_host_key.pub -D libpkcs11.so -I certificate_ID host_key.pub
```

In all cases, *certificate_ID* is a "key identifier" that is logged by the server when the certificate is used for authentication.

Certificates may be configured to be valid only for a set of users or host names, the principals. By default, generated certificates are valid for all users or hosts. To generate a certificate for a specified set of principals, use a comma separated list with the **-n** option as follows:

```
ssh-keygen -s ca_user_key.pub -D libpkcs11.so -I certificate_ID -n user1,user2 id_rsa.pub
```

and for hosts:

```
ssh-keygen -s ca_host_key.pub -D libpkcs11.so -I certificate_ID -h -n host.domain
ssh_host_rsa_key.pub
```

Additional limitations on the validity and use of user certificates may be specified through certificate options. A certificate option may disable features of the SSH session, may be valid only when presented from particular source addresses or may force the use of a specific command. For a list of valid certificate options, see the **ssh-keygen(1)** manual page for the **-O** option.

Certificates may be defined to be valid for a specific lifetime. The **-V** option allows specifying a certificates start and end times. For example:

```
ssh-keygen -s ca_user_key -I certificate_ID id_rsa.pub -V "-1w:+54w5d"
```

A certificate that is presented at a time outside this range will not be considered valid. By default, certificates are valid indefinitely starting from UNIX Epoch.

8.3.7. Viewing an SSH CA Certificate

To view a certificate, use the **-L** to list the contents. For example, for a user's certificate:


```
~]$ ssh-keygen -L -f ~/.ssh/id_rsa-cert.pub
/home/user1/.ssh/id_rsa-cert.pub:
    Type: ssh-rsa-cert-v01@openssh.com user certificate
    Public key: RSA-CERT 3c:9d:42:ed:65:b6:0f:18:bf:52:77:c6:02:0e:e5:86
    Signing CA: RSA b1:8e:0b:ce:fe:1b:67:59:f1:74:cd:32:af:5f:c6:e8
    Key ID: "user1"
    Serial: 0
    Valid: from 2015-05-27T00:09:16 to 2016-06-09T00:09:16
    Principals:
        user1
    Critical Options: (none)
    Extensions:
        permit-X11-forwarding
        permit-agent-forwarding
        permit-port-forwarding
        permit-pty
        permit-user-rc
```

To view a host certificate:

```
~]# ssh-keygen -L -f /etc/ssh/ssh_host_rsa_key-cert.pub
/etc/ssh/ssh_host_rsa_key-cert.pub:
    Type: ssh-rsa-cert-v01@openssh.com host certificate
    Public key: RSA-CERT 1d:71:61:50:05:9b:ec:64:34:27:a5:cc:67:24:03:23
    Signing CA: RSA e4:d5:d1:4f:6b:fd:a2:e3:4e:5a:73:52:91:0b:b7:7a
    Key ID: "host_name"
    Serial: 0
    Valid: from 2015-05-26T17:19:01 to 2016-06-08T17:19:01
    Principals:
        host_name.example.com
    Critical Options: (none)
    Extensions: (none)
```

8.3.8. Revoking an SSH CA Certificate

If a certificate is stolen, it should be revoked. Although OpenSSH does not provide a mechanism to distribute the revocation list it is still easier to create the revocation list and distribute it by other means then to change the CA keys and all host and user certificates previously created and distributed.

Keys can be revoked by adding them to the **revoked_keys** file and specifying the file name in the **sshd_config** file as follows:

```
RevokedKeys /etc/ssh/revoked_keys
```

Note that if this file is not readable, then public key authentication will be refused for all users.

A new key revocation list can be generated as follows:

```
~]$ ssh-keygen -kf /etc/ssh/revoked_keys -z 1 ~/.ssh/id_rsa.pub
```

To add lines to the list, use the **-u** option to update the list:

```
ssh-keygen -ukf /etc/ssh/revoked_keys -z integer ~/.ssh/id_rsa.pub
```

where *integer* is the line number.

To test if a key has been revoked, query the revocation list for the presence of the key. Use a command as follows:


```
ssh-keygen -Qf /etc/ssh/revoked_keys ~/.ssh/id_rsa.pub
```

A user can revoke a CA certificate by changing the **cert-authority** directive to **revoke** in the **known_hosts** file.

8.4. OpenSSH Clients



Make sure you have relevant packages installed

To connect to an OpenSSH server from a client machine, you must have the *openssh-clients* package installed. See [Section 6.2.4, “Installing Packages”](#) for more information on how to install new packages in Fedora 26.

8.4.1. Using the ssh Utility

The **ssh** utility allows you to log in to a remote machine and execute commands there. It is a secure replacement for the **rlogin**, **rsh**, and **telnet** programs.

Similarly to the **telnet** command, log in to a remote machine by using the following command:

```
ssh hostname
```

For example, to log in to a remote machine named `penguin.example.com`, type the following at a shell prompt:

```
~]$ ssh penguin.example.com
```

This will log you in with the same user name you are using on the local machine. If you want to specify a different user name, use a command in the following form:

```
ssh username@hostname
```

For example, to log in to `penguin.example.com` as `USER`, type:

```
~]$ ssh USER@penguin.example.com
```

The first time you initiate a connection, you will be presented with a message similar to this:

```
The authenticity of host 'penguin.example.com' can't be established.  
ECDSA key fingerprint is 256 da:24:43:0b:2e:c1:3f:a1:84:13:92:01:52:b4:84:ff.  
Are you sure you want to continue connecting (yes/no)?
```

Users should always check if the fingerprint is correct before answering the question in this dialog. The user can ask the administrator of the server to confirm the key is correct. This should be done in a secure and previously agreed way. If the user has access to the server's host keys, the fingerprint can be checked by using the **ssh-keygen** command as follows:


```
~]# ssh-keygen -l -f /etc/ssh/ssh_host_ecdsa_key.pub
256 da:24:43:0b:2e:c1:3f:a1:84:13:92:01:52:b4:84:ff (ECDSA)
```

Type **yes** to accept the key and confirm the connection. You will see a notice that the server has been added to the list of known hosts, and a prompt asking for your password:

```
Warning: Permanently added 'penguin.example.com' (ECDSA) to the list of known hosts.
USER@penguin.example.com's password:
```



Updating the host key of an SSH server

If the SSH server's host key changes, the client notifies the user that the connection cannot proceed until the server's host key is deleted from the `~/.ssh/known_hosts` file. Before doing this, however, contact the system administrator of the SSH server to verify the server is not compromised.

To remove a key from the `~/.ssh/known_hosts` file, issue a command as follows:

```
~]# ssh-keygen -R penguin.example.com
# Host penguin.example.com found: line 15 type ECDSA
/home/USER/.ssh/known_hosts updated.
Original contents retained as /home/USER/.ssh/known_hosts.old
```

After entering the password, you will be provided with a shell prompt for the remote machine.

Alternatively, the **ssh** program can be used to execute a command on the remote machine without logging in to a shell prompt:

```
ssh [username@]hostname command
```

For example, the `/etc/redhat-release` file provides information about the Fedora version. To view the contents of this file on `penguin.example.com`, type:

```
~]$ ssh USER@penguin.example.com cat /etc/redhat-release
USER@penguin.example.com's password:
Fedora release 20 (Heisenbug)
```

After you enter the correct password, the user name will be displayed, and you will return to your local shell prompt.

8.4.2. Using the scp Utility

scp can be used to transfer files between machines over a secure, encrypted connection. In its design, it is very similar to **rcp**.

To transfer a local file to a remote system, use a command in the following form:

```
scp localfile username@hostname:remotefile
```


For example, if you want to transfer **taglist.vim** to a remote machine named `penguin.example.com`, type the following at a shell prompt:

```
~]$ scp taglist.vim USER@penguin.example.com:~/.vim/plugin/taglist.vim
USER@penguin.example.com's password:
taglist.vim                                100% 144KB 144.5KB/s 00:00
```

Multiple files can be specified at once. To transfer the contents of `~/.vim/plugin/` to the same directory on the remote machine `penguin.example.com`, type the following command:

```
~]$ scp ~/.vim/plugin/* USER@penguin.example.com:~/.vim/plugin/
USER@penguin.example.com's password:
closetag.vim                                100% 13KB 12.6KB/s 00:00
snippetsEmu.vim                            100% 33KB 33.1KB/s 00:00
taglist.vim                                100% 144KB 144.5KB/s 00:00
```

To transfer a remote file to the local system, use the following syntax:

```
scp username@hostname:remotefile localfile
```

For instance, to download the `~/.vimrc` configuration file from the remote machine, type:

```
~]$ scp USER@penguin.example.com:~/.vimrc ~/.vimrc
USER@penguin.example.com's password:
~/.vimrc                                    100% 2233 2.2KB/s 00:00
```

8.4.3. Using the **sftp** Utility

The **sftp** utility can be used to open a secure, interactive FTP session. In its design, it is similar to **ftp** except that it uses a secure, encrypted connection.

To connect to a remote system, use a command in the following form:

```
sftp username@hostname
```

For example, to log in to a remote machine named `penguin.example.com` with `USER` as a user name, type:

```
~]$ sftp USER@penguin.example.com
USER@penguin.example.com's password:
Connected to penguin.example.com.
sftp>
```

After you enter the correct password, you will be presented with a prompt. The **sftp** utility accepts a set of commands similar to those used by **ftp** (see [Table 8.3, “A selection of available sftp commands”](#)).

Table 8.3. A selection of available **sftp** commands

Command	Description
ls [<i>directory</i>]	List the content of a remote <i>directory</i> . If none is supplied, a current working directory is used by default.
cd <i>directory</i>	Change the remote working directory to <i>directory</i> .
mkdir <i>directory</i>	Create a remote <i>directory</i> .

Command	Description
<code>rmdir path</code>	Remove a remote <i>directory</i> .
<code>put localfile [remotefile]</code>	Transfer <i>localfile</i> to a remote machine.
<code>get remotefile [localfile]</code>	Transfer <i>remotefile</i> from a remote machine.

For a complete list of available commands, see the **sftp(1)** manual page.

8.5. More Than a Secure Shell

A secure command line interface is just the beginning of the many ways SSH can be used. Given the proper amount of bandwidth, X11 sessions can be directed over an SSH channel. Or, by using TCP/IP forwarding, previously insecure port connections between systems can be mapped to specific SSH channels.

8.5.1. X11 Forwarding

To open an X11 session over an SSH connection, use a command in the following form:

```
ssh -Y username@hostname
```

For example, to log in to a remote machine named `penguin.example.com` with `USER` as a user name, type:

```
~]$ ssh -Y USER@penguin.example.com
USER@penguin.example.com's password:
```

When an X program is run from the secure shell prompt, the SSH client and server create a new secure channel, and the X program data is sent over that channel to the client machine transparently.

Note that the X Window system must be installed on the remote system before X11 forwarding can take place. Enter the following command as `root` to install the X11 package group:

```
~]# dnf group install "X Window System"
```

X11 forwarding can be very useful. For example, X11 forwarding can be used to create a secure, interactive session of the **Print Settings** utility. To do this, connect to the server using **ssh** and type:

```
~]$ system-config-printer &
```

The **Print Settings** tool will appear, allowing the remote user to safely configure printing on the remote system.

8.5.2. Port Forwarding

SSH can secure otherwise insecure TCP/IP protocols via port forwarding. When using this technique, the SSH server becomes an encrypted conduit to the SSH client.

Port forwarding works by mapping a local port on the client to a remote port on the server. SSH can map any port from the server to any port on the client. Port numbers do not need to match for this technique to work.



Using reserved port numbers

Setting up port forwarding to listen on ports below 1024 requires root level access.

To create a TCP/IP port forwarding channel which listens for connections on the localhost, use a command in the following form:

```
ssh -L local-port:remote-hostname:remote-port username@hostname
```

For example, to check email on a server called mail.example.com using POP3 through an encrypted connection, use the following command:

```
~]$ ssh -L 1100:mail.example.com:110 mail.example.com
```

Once the port forwarding channel is in place between the client machine and the mail server, direct a POP3 mail client to use port **1100** on the localhost to check for new email. Any requests sent to port **1100** on the client system will be directed securely to the mail.example.com server.

If mail.example.com is not running an SSH server, but another machine on the same network is, SSH can still be used to secure part of the connection. However, a slightly different command is necessary:

```
~]$ ssh -L 1100:mail.example.com:110 other.example.com
```

In this example, POP3 requests from port **1100** on the client machine are forwarded through the SSH connection on port **22** to the SSH server, other.example.com. Then, other.example.com connects to port **110** on mail.example.com to check for new email. Note that when using this technique, only the connection between the client system and other.example.com SSH server is secure.

Port forwarding can also be used to get information securely through network firewalls. If the firewall is configured to allow SSH traffic via its standard port (that is, port 22) but blocks access to other ports, a connection between two hosts using the blocked ports is still possible by redirecting their communication over an established SSH connection.



A connection is only as secure as a client system

Using port forwarding to forward connections in this manner allows any user on the client system to connect to that service. If the client system becomes compromised, the attacker also has access to forwarded services.

System administrators concerned about port forwarding can disable this functionality on the server by specifying a **No** parameter for the **AllowTcpForwarding** line in **/etc/ssh/sshd_config** and restarting the **sshd** service.

8.6. Additional Resources

For more information on how to configure or connect to an OpenSSH server on Fedora, see the resources listed below.

Installed Documentation

- `sshd(8)` — The manual page for the `sshd` daemon documents available command line options and provides a complete list of supported configuration files and directories.
- `ssh(1)` — The manual page for the **ssh** client application provides a complete list of available command line options and supported configuration files and directories.
- `scp(1)` — The manual page for the **scp** utility provides a more detailed description of this utility and its usage.
- `sftp(1)` — The manual page for the **sftp** utility.
- `ssh-keygen(1)` — The manual page for the **ssh-keygen** utility documents in detail how to use it to generate, manage, and convert authentication keys used by **ssh**.
- `ssh_config(5)` — The manual page named `ssh_config` documents available SSH client configuration options.
- `sshd_config(5)` — The manual page named `sshd_config` provides a full description of available SSH daemon configuration options.

Online Documentation

- [OpenSSH Home Page](http://www.openssh.com/)³ — The OpenSSH home page containing further documentation, frequently asked questions, links to the mailing lists, bug reports, and other useful resources.
- [OpenSSL Home Page](http://www.openssl.org/)⁴ — The OpenSSL home page containing further documentation, frequently asked questions, links to the mailing lists, and other useful resources.

³ <http://www.openssh.com/>

⁴ <http://www.openssl.org/>

TigerVNC

TigerVNC (Tiger Virtual Network Computing) is a system for graphical desktop sharing which allows you to remotely control other computers.

TigerVNC works on the client-server network: a **server** shares its output (`vncserver`) and a **client** (`vncviewer`) connects to the server.

Note

Unlike in Fedora 15 and Red Hat Enterprise Linux 6, TigerVNC in Fedora uses the `systemd` system management daemon for its configuration. The `/etc/sysconfig/vncserver` configuration file has been replaced by `/etc/systemd/system/vncserver@.service`.

9.1. VNC Server

`vncserver` is a utility which starts a VNC (Virtual Network Computing) desktop. It runs **Xvnc** with appropriate options and starts a window manager on the VNC desktop. `vncserver` allows users to run separate sessions in parallel on a machine which can then be accessed by any number of clients from anywhere.

9.1.1. Installing VNC Server

To install the **TigerVNC** server, issue the following command as root:

```
~]# dnf install tigervnc-server
```

9.1.2. Configuring VNC Server

Procedure 9.1. Configuring the first VNC connection

1. A configuration file named `/etc/systemd/system/vncserver@.service` is required. To create this file, copy the `/lib/systemd/system/vncserver@.service` file as root:

```
~]# cp /lib/systemd/system/vncserver@.service /etc/systemd/system/vncserver@.service
```

There is no need to include the display number in the file name because `systemd` automatically creates the appropriately named instance in memory on demand, replacing `'%i'` in the service file by the display number. For a single user it is not necessary to rename the file. For multiple users, a uniquely named service file for each user is required, for example, by adding the user name to the file name in some way. See [Section 9.1.2.1, “Configuring VNC Server for Two Users”](#) for details.

2. Edit `/etc/systemd/system/vncserver@.service`, replacing `USER` with the actual user name. Leave the remaining lines of the file unmodified. The **-geometry** argument specifies the size of the VNC desktop to be created; by default, it is set to **1024x768**.

```
ExecStart=/sbin/runuser -l USER -c "/usr/bin/vncserver %i -geometry 1280x1024"
PIDFile=/home/USER/.vnc/%H%i.pid
```


3. Save the changes.
4. To make the changes take effect immediately, issue the following command:

```
~]# systemctl daemon-reload
```

5. Set the password for the user or users defined in the configuration file. Note that you need to switch from root to *USER* first.

```
~]# su - USER
~]$ vncpasswd
Password:
Verify:
```



Important

The stored password is not encrypted; anyone who has access to the password file can find the plain-text password.

Proceed to [Section 9.1.3, “Starting VNC Server”](#).

9.1.2.1. Configuring VNC Server for Two Users

If you want to configure more than one user on the same machine, create different template-type service files, one for each user.

1. Create two service files, for example **vncserver-*USER_1*@.service** and **vncserver-*USER_2*@.service**. In both these files substitute *USER* with the correct user name.
2. Set passwords for both users:

```
~]$ su - USER_1
~]$ vncpasswd
Password:
Verify:
~]$ su - USER_2
~]$ vncpasswd
Password:
Verify:
```

9.1.3. Starting VNC Server

To start or enable the service, specify the display number directly in the command. The file configured above in [Procedure 9.1, “Configuring the first VNC connection”](#) works as a template, in which **%i** is substituted with the display number by `systemd`. With a valid display number, execute the following command:


```
~]# systemctl start vncserver@:display_number.service
```

You can also enable the service to start automatically at system start. Then, when you log in, vncserver is automatically started. As root, issue a command as follows:

```
~]# systemctl enable vncserver@:display_number.service
```

At this point, other users are able to use a VNC viewer program to connect to the VNC server using the display number and password defined. Provided a graphical desktop is installed, an instance of that desktop will be displayed. It will not be the same instance as that currently displayed on the target machine.

9.1.3.1. Configuring VNC Server for Two Users and Two Different Displays

For the two configured VNC servers, vncserver-USER_1@.service and vncserver-USER_2@.service, you can enable different display numbers. For example, the following commands will cause a VNC server for USER_1 to start on display 3, and a VNC server for USER_2 to start on display 5:

```
~]# systemctl start vncserver-USER_1@:3.service  
~]# systemctl start vncserver-USER_2@:5.service
```

9.1.4. Terminating a VNC Session

Similarly to enabling the vncserver service, you can disable the automatic start of the service at system start:

```
~]# systemctl disable vncserver@:display_number.service
```

Or, when your system is running, you can stop the service by issuing the following command as root:

```
~]# systemctl stop vncserver@:display_number.service
```

9.2. VNC Viewer

vncviewer is the program which shows the shared graphical user interfaces and controls the server.

For operating the vncviewer, there is a pop-up menu containing entries which perform various actions such as switching in and out of full-screen mode or quitting the viewer. Alternatively, you can operate vncviewer through the terminal. Enter **vncviewer -h** on the command line to list vncviewer's parameters.

9.2.1. Installing VNC Viewer

To install the **TigerVNC** client, **vncviewer**>, issue the following command as root:

```
~]# dnf install tigervnc
```


9.2.2. Connecting to VNC Server

Once the VNC server is configured, you can connect to it from any VNC viewer. In order to do so, issue the **vncviewer** command in the following format:

```
vncviewer address:port_number
```

Where *address* is an IP or host name.

Example 9.1. One Client Connecting to VNC Server

With the IP address 192.168.0.4 and display number 3 the command looks as follows:

```
~]$ vncviewer 192.168.0.4:3
```

9.2.2.1. Configuring the Firewall for VNC

When using a non-encrypted connection, **firewalld** might block the connection. To allow **firewalld** to pass the VNC packets, you can open specific ports to TCP traffic. When using the **-via** option, traffic is redirected over SSH which is enabled by default in **firewalld**.

Note

The default port of VNC server is 5900. To reach the port through which a remote desktop will be accessible, sum the default port and the user's assigned display number. For example, for the second port: 2 + 5900 = 5902.

For displays **0** to **3**, make use of **firewalld**'s support for the VNC service by means of the **service** option as described below. Note that for display numbers greater than **3**, the corresponding ports will have to be opened specifically as explained in [Procedure 9.3, "Opening Ports in firewalld"](#).

Procedure 9.2. Enabling VNC Service in firewalld

1. Run the following command to see the information concerning **firewalld** settings:

```
~]$ firewall-cmd --list-all
```

2. To allow all VNC connections from a specific address, use a command as follows:

```
~]# firewall-cmd --add-rich-rule='rule family="ipv4" source address="192.168.122.116"
service name=vnc-server accept'
success
```

See the [Red Hat Enterprise Linux 7 Security Guide](#)¹ for more information on the use of firewall rich language commands.

3. To verify the above settings, use a command as follows:

¹ https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/


```
~]# firewall-cmd --list-all
public (default, active)
  interfaces: bond0 bond0.192
  sources:
  services: dhcpv6-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
  rule family="ipv4" source address="192.168.122.116" service name="vnc-server" accept
```

To open a specific port or range of ports make use of the **--add-port** option to the **firewall-cmd** command Line tool. For example, VNC display **4** requires port **5904** to be opened for TCP traffic.

Procedure 9.3. Opening Ports in firewalld

1. To open a port for TCP traffic in the public zone, issue a command as root as follows:

```
~]# firewall-cmd --zone=public --add-port=5904/tcp
success
```

2. To view the ports that are currently open for the public zone, issue a command as follows:

```
~]# firewall-cmd --zone=public --list-ports
5904/tcp
```

A port can be removed using the **firewall-cmd --zone=zone --remove-port=number/protocol** command.

For more information on opening and closing ports in firewalld, see the [Red Hat Enterprise Linux 7 Security Guide](#)².

9.2.3. Connecting to VNC Server Using SSH

VNC is a clear text network protocol with no security against possible attacks on the communication. To make the communication secure, you can encrypt your server-client connection by using the **-via** option. This will create an SSH tunnel between the VNC server and the client.

The format of the command to encrypt a VNC server-client connection is as follows:

```
~]$ vncviewer -via user@host:display_number
```

Example 9.2. Using the -via Option

1. To connect to a VNC server using SSH, enter a command as follows:

```
~]$ vncviewer -via USER_2@192.168.2.101:3
```

2. When you are prompted to, type the password, and confirm by pressing **Enter**.

² https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/

3. A window with a remote desktop appears on your screen.

Restricting VNC Access

If you prefer only encrypted connections, you can prevent unencrypted connections altogether by using the **-localhost** option in the `systemd.service` file, the `ExecStart` line:

```
ExecStart=/sbin/runuser -l user -c "/usr/bin/vncserver -localhost %i"
```

This will stop `vncserver` from accepting connections from anything but the local host and port-forwarded connections sent using SSH as a result of the **-via** option.

For more information on using SSH, see [Chapter 8, OpenSSH](#).

9.3. Additional Resources

For more information about TigerVNC, see the resources listed below.

Installed Documentation

- **vncserver(1)** — The VNC server manual pages.
- **vncviewer(1)** — The VNC viewer manual pages.
- **vncpasswd(1)** — The VNC password manual pages.

Part IV. Servers

This part discusses various topics related to servers such as how to set up a web server or share files and directories over a network.

Web Servers

A *web server* is a network service that serves content to a client over the web. This typically means web pages, but any other documents can be served as well. Web servers are also known as HTTP servers, as they use the *hypertext transport protocol* (HTTP).

10.1. The Apache HTTP Server

The web server available in Fedora is the Apache HTTP server daemon, `httpd`, an open source web server developed by the [Apache Software Foundation](http://www.apache.org/)¹. In Fedora 19 the Apache server was updated to **Apache HTTP Server 2.4**. This section describes the basic configuration of the `httpd` service, and covers some advanced topics such as adding server modules, setting up virtual hosts, or configuring the secure HTTP server.

There are important differences between the Apache HTTP Server 2.4 and version 2.2, and if you are upgrading from a release prior to Fedora 19, you will need to update the `httpd` service configuration accordingly. This section reviews some of the newly added features, outlines important changes, and guides you through the update of older configuration files.

10.1.1. Notable Changes

The Apache HTTP Server version 2.4 has the following changes compared to version 2.2:

httpd Service Control

With the migration away from SysV init scripts, server administrators should switch to using the **`apachectl`** and **`systemctl`** commands to control the service, in place of the **`service`** command. The following examples are specific to the `httpd` service.

The command:

```
service httpd graceful
```

is replaced by

```
apachectl graceful
```

The `systemd` unit file for `httpd` has different behavior from the init script as follows:

- A graceful restart is used by default when the service is reloaded.
- A graceful stop is used by default when the service is stopped.

The command:

```
service httpd configtest
```

is replaced by

```
apachectl configtest
```

¹ <http://www.apache.org/>

Private /tmp

To enhance system security, the `systemd` unit file runs the `httpd` daemon using a private `/tmp` directory, separate to the system `/tmp` directory.

Configuration Layout

Configuration files which load modules are now placed in the `/etc/httpd/conf.modules.d` directory. Packages that provide additional loadable modules for `httpd`, such as `php`, will place a file in this directory. Any configuration files in the `conf.modules.d` directory are processed before the main body of `httpd.conf`. Configuration files in the `/etc/httpd/conf.d` directory are now processed after the main body of `httpd.conf`.

Some additional configuration files are provided by the `httpd` package itself:

- `/etc/httpd/conf.d/autoindex.conf` — This configures `mod_autoindex` directory indexing.
- `/etc/httpd/conf.d/userdir.conf` — This configures access to user directories, for example, `http://example.com/~username/`; such access is disabled by default for security reasons.
- `/etc/httpd/conf.d/welcome.conf` — As in previous releases, this configures the welcome page displayed for `http://localhost/` when no content is present.

Default Configuration

A minimal `httpd.conf` file is now provided by default. Many common configuration settings, such as `Timeout` or `KeepAlive` are no longer explicitly configured in the default configuration; hard-coded settings will be used instead, by default. The hard-coded default settings for all configuration directives are specified in the manual. See [the section called “Installable Documentation”](#) for more information.

Incompatible Syntax Changes

If migrating an existing configuration from `httpd 2.2` to `httpd 2.4`, a number of backwards-incompatible changes to the `httpd` configuration syntax were made which will require changes. See the following Apache document for more information on upgrading <http://httpd.apache.org/docs/2.4/upgrading.html>

Processing Model

In previous releases of Fedora, different *multi-processing models* (MPM) were made available as different `httpd` binaries: the forked model, “prefork”, as `/usr/sbin/httpd`, and the thread-based model “worker” as `/usr/sbin/httpd.worker`.

In Fedora 26, only a single `httpd` binary is used, and three MPMs are available as loadable modules: worker, prefork (default), and event. Edit the configuration file `/etc/httpd/conf.modules.d/00-mpm.conf` as required, by adding and removing the comment character `#` so that only one of the three MPM modules is loaded.

Packaging Changes

The LDAP authentication and authorization modules are now provided in a separate sub-package, `mod_ldap`. The new module `mod_session` and associated helper modules are provided in a new sub-package, `mod_session`. The new modules `mod_proxy_html` and `mod_xml2enc` are provided in a new sub-package, `mod_proxy_html`.

Packaging Filesystem Layout

The `/var/cache/mod_proxy/` directory is no longer provided; instead, the `/var/cache/httpd/` directory is packaged with a `proxy` and `ssl` subdirectory.

Packaged content provided with `httpd` has been moved from `/var/www/` to `/usr/share/httpd/`:

- `/usr/share/httpd/icons/` — The directory containing a set of icons used with directory indices, previously contained in `/var/www/icons/`, has moved to `/usr/share/httpd/icons`. Available at `http://localhost/icons/` in the default configuration; the location and the availability of the icons is configurable in the `/etc/httpd/conf.d/autoindex.conf` file.
- `/usr/share/httpd/manual/` — The `/var/www/manual/` has moved to `/usr/share/httpd/manual/`. This directory, contained in the `httpd-manual` package, contains the HTML version of the manual for `httpd`. Available at `http://localhost/manual/` if the package is installed, the location and the availability of the manual is configurable in the `/etc/httpd/conf.d/manual.conf` file.
- `/usr/share/httpd/error/` — The `/var/www/error/` has moved to `/usr/share/httpd/error/`. Custom multi-language HTTP error pages. Not configured by default, the example configuration file is provided at `/usr/share/doc/httpd-VERSION/httpd-multilang-errordoc.conf`.

Authentication, Authorization and Access Control

The configuration directives used to control authentication, authorization and access control have changed significantly. Existing configuration files using the **Order**, **Deny** and **Allow** directives should be adapted to use the new **Require** syntax. See the following Apache document for more information <http://httpd.apache.org/docs/2.4/howto/auth.html>

suexec

To improve system security, the **suexec** binary is no longer installed as if by the root user; instead, it has file system capability bits set which allow a more restrictive set of permissions. In conjunction with this change, the **suexec** binary no longer uses the `/var/log/httpd/suexec.log` logfile. Instead, log messages are sent to **syslog**; by default these will appear in the `/var/log/secure` log file.

Module Interface

Third-party binary modules built against **httpd 2.2** are not compatible with **httpd 2.4** due to changes to the `httpd` module interface. Such modules will need to be adjusted as necessary for the **httpd 2.4** module interface, and then rebuilt. A detailed list of the API changes in version **2.4** is available here: http://httpd.apache.org/docs/2.4/developer/new_api_2_4.html.

The **apxs** binary used to build modules from source has moved from `/usr/sbin/apxs` to `/usr/bin/apxs`.

Removed modules

List of `httpd` modules removed in Fedora 26:

`mod_auth_mysql`, `mod_auth_pgsq`

httpd 2.4 provides SQL database authentication support internally in the `mod_authn_dbd` module.

`mod_perl`

mod_perl is not officially supported with **httpd 2.4** by upstream.

`mod_authz_ldap`

httpd 2.4 provides LDAP support in sub-package `mod_ldap` using `mod_authnz_ldap`.

10.1.2. Updating the Configuration

To update the configuration files from the Apache HTTP Server version 2.2, take the following steps:

1. Make sure all module names are correct, since they may have changed. Adjust the **LoadModule** directive for each module that has been renamed.
2. Recompile all third party modules before attempting to load them. This typically means authentication and authorization modules.
3. If you use the `mod_userdir` module, make sure the **UserDir** directive indicating a directory name (typically **public_html**) is provided.
4. If you use the Apache HTTP Secure Server, edit the `/etc/httpd/conf.d/ssl.conf` to enable the Secure Sockets Layer (SSL) protocol.

Note that you can check the configuration for possible errors by using the following command:

```
~]# apachectl configtest
Syntax OK
```

For more information on upgrading the Apache HTTP Server configuration from version 2.2 to 2.4, see <http://httpd.apache.org/docs/2.4/upgrading.html>.

10.1.3. Running the httpd Service

This section describes how to start, stop, restart, and check the current status of the Apache HTTP Server. To be able to use the `httpd` service, make sure you have the `httpd` installed. You can do so by using the following command:

```
~]# dnf install httpd
```

For more information on the concept of targets and how to manage system services in Fedora in general, see [Chapter 7, Services and Daemons](#).

10.1.3.1. Starting the Service

To run the `httpd` service, type the following at a shell prompt as root:

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

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

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



Using the secure server

If running the Apache HTTP Server as a secure server, a password may be required after the machine boots if using an encrypted private SSL key.

10.1.3.2. Stopping the Service

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

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

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

```
~]# systemctl disable httpd.service  
rm '/etc/systemd/system/multi-user.target.wants/httpd.service'
```

10.1.3.3. Restarting the Service

There are three different ways to restart a running `httpd` service:

1. To restart the service completely, enter the following command as root:

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

This stops the running `httpd` service and immediately starts it again. Use this command after installing or removing a dynamically loaded module such as PHP.

2. To only reload the configuration, as root, type:

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

This causes the running `httpd` service to reload its configuration file. Any requests currently being processed will be interrupted, which may cause a client browser to display an error message or render a partial page.

3. To reload the configuration without affecting active requests, enter the following command as root:

```
~]# apachectl graceful
```

This causes the running `httpd` service to reload its configuration file. Any requests currently being processed will continue to use the old configuration.

10.1.3.4. Verifying the Service Status

To verify that the `httpd` service is running, type the following at a shell prompt:

```
~]# systemctl is-active httpd.service  
active
```

10.1.4. Editing the Configuration Files

When the `httpd` service is started, by default, it reads the configuration from locations that are listed in [Table 10.1, “The `httpd` service configuration files”](#).

Table 10.1. The `httpd` service configuration files

Path	Description
<code>/etc/httpd/conf/httpd.conf</code>	The main configuration file.
<code>/etc/httpd/conf.d/</code>	An auxiliary directory for configuration files that are included in the main configuration file.

Although the default configuration should be suitable for most situations, it is a good idea to become at least familiar with some of the more important configuration options. Note that for any changes to take effect, the web server has to be restarted first. See [Section 10.1.3.3, “Restarting the Service”](#) for more information on how to restart the `httpd` service.

To check the configuration for possible errors, type the following at a shell prompt:

```
~]# apachectl configtest
Syntax OK
```

To make the recovery from mistakes easier, it is recommended that you make a copy of the original file before editing it.

10.1.4.1. Common `httpd.conf` Directives

The following directives are commonly used in the `/etc/httpd/conf/httpd.conf` configuration file:

<Directory>

The **<Directory>** directive allows you to apply certain directives to a particular directory only. It takes the following form:

```
<Directory directory>
    directive
    ...
</Directory>
```

The *directory* can be either a full path to an existing directory in the local file system, or a wildcard expression.

This directive can be used to configure additional **cgi-bin** directories for server-side scripts located outside the directory that is specified by **ScriptAlias**. In this case, the **ExecCGI** and **AddHandler** directives must be supplied, and the permissions on the target directory must be set correctly (that is, **0755**).

Example 10.1. Using the `<Directory>` directive

```
<Directory /var/www/html>
    Options Indexes FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```


<IfDefine>

The **IfDefine** directive allows you to use certain directives only when a particular parameter is supplied on the command line. It takes the following form:

```
<IfDefine [!]parameter>
    directive
...
</IfDefine>
```

The *parameter* can be supplied at a shell prompt using the **-Dparameter** command line option (for example, **httpd -DEnableHome**). If the optional exclamation mark (that is, **!**) is present, the enclosed directives are used only when the parameter is *not* specified.

Example 10.2. Using the <IfDefine> directive

```
<IfDefine EnableHome>
    UserDir public_html
</IfDefine>
```

<IfModule>

The **<IfModule>** directive allows you to use certain directive only when a particular module is loaded. It takes the following form:

```
<IfModule [!]module>
    directive
...
</IfModule>
```

The *module* can be identified either by its name, or by the file name. If the optional exclamation mark (that is, **!**) is present, the enclosed directives are used only when the module is *not* loaded.

Example 10.3. Using the <IfModule> directive

```
<IfModule mod_disk_cache.c>
    CacheEnable disk /
    CacheRoot /var/cache/mod_proxy
</IfModule>
```

<Location>

The **<Location>** directive allows you to apply certain directives to a particular URL only. It takes the following form:

```
<Location url>
    directive
...
</Location>
```

The *url* can be either a path relative to the directory specified by the **DocumentRoot** directive (for example, **/server-info**), or an external URL such as **http://example.com/server-info**.

Example 10.4. Using the <Location> directive

```
<Location /server-info>
```



```

    SetHandler server-info
    Order deny,allow
    Deny from all
    Allow from .example.com
  </Location>

```

<Proxy>

The **<Proxy>** directive allows you to apply certain directives to the proxy server only. It takes the following form:

```

<Proxy pattern>
  directive
  ...
</Proxy>

```

The *pattern* can be an external URL, or a wildcard expression (for example, **http://example.com/***).

Example 10.5. Using the <Proxy> directive

```

<Proxy *>
  Order deny,allow
  Deny from all
  Allow from .example.com
</Proxy>

```

<VirtualHost>

The **<VirtualHost>** directive allows you apply certain directives to particular virtual hosts only. It takes the following form:

```

<VirtualHost address[:port]...>
  directive
  ...
</VirtualHost>

```

The *address* can be an IP address, a fully qualified domain name, or a special form as described in [Table 10.2, “Available <VirtualHost> options”](#).

Table 10.2. Available <VirtualHost> options

Option	Description
*	Represents all IP addresses.
default	Represents unmatched IP addresses.

Example 10.6. Using the <VirtualHost> directive

```

<VirtualHost *:80>
  ServerAdmin webmaster@penguin.example.com
  DocumentRoot /www/docs/penguin.example.com
  ServerName penguin.example.com
  ErrorLog logs/penguin.example.com-error_log
  CustomLog logs/penguin.example.com-access_log common
</VirtualHost>

```


AccessFileName

The **AccessFileName** directive allows you to specify the file to be used to customize access control information for each directory. It takes the following form:

```
AccessFileName filename...
```

The *filename* is a name of the file to look for in the requested directory. By default, the server looks for **.htaccess**.

For security reasons, the directive is typically followed by the **Files** tag to prevent the files beginning with **.ht** from being accessed by web clients. This includes the **.htaccess** and **.htpasswd** files.

Example 10.7. Using the AccessFileName directive

```
AccessFileName .htaccess

<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
    Satisfy All
</Files>
```

Action

The **Action** directive allows you to specify a CGI script to be executed when a certain media type is requested. It takes the following form:

```
Action content-type path
```

The *content-type* has to be a valid MIME type such as **text/html**, **image/png**, or **application/pdf**. The *path* refers to an existing CGI script, and must be relative to the directory specified by the **DocumentRoot** directive (for example, **/cgi-bin/process-image.cgi**).

Example 10.8. Using the Action directive

```
Action image/png /cgi-bin/process-image.cgi
```

AddDescription

The **AddDescription** directive allows you to specify a short description to be displayed in server-generated directory listings for a given file. It takes the following form:

```
AddDescription "description" filename...
```

The *description* should be a short text enclosed in double quotes (that is, "). The *filename* can be a full file name, a file extension, or a wildcard expression.

Example 10.9. Using the AddDescription directive

```
AddDescription "GZIP compressed tar archive" .tgz
```


AddEncoding

The **AddEncoding** directive allows you to specify an encoding type for a particular file extension. It takes the following form:

```
AddEncoding encoding extension...
```

The *encoding* has to be a valid MIME encoding such as **x-compress**, **x-gzip**, etc. The *extension* is a case sensitive file extension, and is conventionally written with a leading dot (for example, **.gz**).

This directive is typically used to instruct web browsers to decompress certain file types as they are downloaded.

Example 10.10. Using the AddEncoding directive

```
AddEncoding x-gzip .gz .tgz
```

AddHandler

The **AddHandler** directive allows you to map certain file extensions to a selected handler. It takes the following form:

```
AddHandler handler extension...
```

The *handler* has to be a name of previously defined handler. The *extension* is a case sensitive file extension, and is conventionally written with a leading dot (for example, **.cgi**).

This directive is typically used to treat files with the **.cgi** extension as CGI scripts regardless of the directory they are in. Additionally, it is also commonly used to process server-parsed HTML and image-map files.

Example 10.11. Using the AddHandler option

```
AddHandler cgi-script .cgi
```

AddIcon

The **AddIcon** directive allows you to specify an icon to be displayed for a particular file in server-generated directory listings. It takes the following form:

```
AddIcon path pattern...
```

The *path* refers to an existing icon file, and must be relative to the directory specified by the **DocumentRoot** directive (for example, **/icons/folder.png**). The *pattern* can be a file name, a file extension, a wildcard expression, or a special form as described in the following table:

Table 10.3. Available AddIcon options

Option	Description
^^DIRECTORY^^	Represents a directory.
^^BLANKICON^^	Represents a blank line.

Example 10.12. Using the AddIcon directive

```
AddIcon /icons/text.png .txt README
```

AddIconByEncoding

The **AddIconByEncoding** directive allows you to specify an icon to be displayed for a particular encoding type in server-generated directory listings. It takes the following form:

```
AddIconByEncoding path encoding...
```

The *path* refers to an existing icon file, and must be relative to the directory specified by the **DocumentRoot** directive (for example, **/icons/compressed.png**). The *encoding* has to be a valid MIME encoding such as **x-compress**, **x-gzip**, etc.

Example 10.13. Using the AddIconByEncoding directive

```
AddIconByEncoding /icons/compressed.png x-compress x-gzip
```

AddIconByType

The **AddIconByType** directive allows you to specify an icon to be displayed for a particular media type in server-generated directory listings. It takes the following form:

```
AddIconByType path content-type...
```

The *path* refers to an existing icon file, and must be relative to the directory specified by the **DocumentRoot** directive (for example, **/icons/text.png**). The *content-type* has to be either a valid MIME type (for example, **text/html** or **image/png**), or a wildcard expression such as **text/***, **image/***, etc.

Example 10.14. Using the AddIconByType directive

```
AddIconByType /icons/video.png video/*
```

AddLanguage

The **AddLanguage** directive allows you to associate a file extension with a specific language. It takes the following form:

```
AddLanguage language extension...
```

The *language* has to be a valid MIME language such as **cs**, **en**, or **fr**. The *extension* is a case sensitive file extension, and is conventionally written with a leading dot (for example, **.cs**).

This directive is especially useful for web servers that serve content in multiple languages based on the client's language settings.

Example 10.15. Using the AddLanguage directive

```
AddLanguage cs .cs .cz
```


AddType

The **AddType** directive allows you to define or override the media type for a particular file extension. It takes the following form:

```
AddType content-type extension...
```

The *content-type* has to be a valid MIME type such as **text/html**, **image/png**, etc. The *extension* is a case sensitive file extension, and is conventionally written with a leading dot (for example, **.cs**).

Example 10.16. Using the AddType directive

```
AddType application/x-gzip .gz .tgz
```

Alias

The **Alias** directive allows you to refer to files and directories outside the default directory specified by the **DocumentRoot** directive. It takes the following form:

```
Alias url-path real-path
```

The *url-path* must be relative to the directory specified by the **DocumentRoot** directive (for example, **/images/**). The *real-path* is a full path to a file or directory in the local file system.

This directive is typically followed by the **Directory** tag with additional permissions to access the target directory. By default, the **/icons/** alias is created so that the icons from **/var/www/icons/** are displayed in server-generated directory listings.

Example 10.17. Using the Alias directive

```
Alias /icons/ /var/www/icons/

<Directory "/var/www/icons">
    Options Indexes MultiViews FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

Allow

The **Allow** directive allows you to specify which clients have permission to access a given directory. It takes the following form:

```
Allow from client...
```

The *client* can be a domain name, an IP address (both full and partial), a *network/netmask* pair, or **all** for all clients.

Example 10.18. Using the Allow directive

```
Allow from 192.168.1.0/255.255.255.0
```


AllowOverride

The **AllowOverride** directive allows you to specify which directives in a **.htaccess** file can override the default configuration. It takes the following form:

```
AllowOverride type...
```

The *type* has to be one of the available grouping options as described in [Table 10.4, “Available AllowOverride options”](#).

Table 10.4. Available AllowOverride options

Option	Description
All	All directives in .htaccess are allowed to override earlier configuration settings.
None	No directive in .htaccess is allowed to override earlier configuration settings.
AuthConfig	Allows the use of authorization directives such as AuthName , AuthType , or Require .
FileInfo	Allows the use of file type, metadata, and <code>mod_rewrite</code> directives such as DefaultType , RequestHeader , or RewriteEngine , as well as the Action directive.
Indexes	Allows the use of directory indexing directives such as AddDescription , AddIcon , or FancyIndexing .
Limit	Allows the use of host access directives, that is, Allow , Deny , and Order .
Options [= <i>option</i> , ...]	Allows the use of the Options directive. Additionally, you can provide a comma-separated list of options to customize which options can be set using this directive.

Example 10.19. Using the AllowOverride directive

```
AllowOverride FileInfo AuthConfig Limit
```

BrowserMatch

The **BrowserMatch** directive allows you to modify the server behavior based on the client's web browser type. It takes the following form:

```
BrowserMatch pattern variable...
```

The *pattern* is a regular expression to match the User-Agent HTTP header field. The *variable* is an environment variable that is set when the header field matches the pattern.

By default, this directive is used to deny connections to specific browsers with known issues, and to disable keepalives and HTTP header flushes for browsers that are known to have problems with these actions.

Example 10.20. Using the BrowserMatch directive

```
BrowserMatch "Mozilla/2" nokeepalive
```


CacheDefaultExpire

The **CacheDefaultExpire** option allows you to set how long to cache a document that does not have any expiration date or the date of its last modification specified. It takes the following form:

```
CacheDefaultExpire time
```

The *time* is specified in seconds. The default option is **3600** (that is, one hour).

Example 10.21. Using the CacheDefaultExpire directive

```
CacheDefaultExpire 3600
```

CacheDisable

The **CacheDisable** directive allows you to disable caching of certain URLs. It takes the following form:

```
CacheDisable path
```

The *path* must be relative to the directory specified by the **DocumentRoot** directive (for example, **/files/**).

Example 10.22. Using the CacheDisable directive

```
CacheDisable /temporary
```

CacheEnable

The **CacheEnable** directive allows you to specify a cache type to be used for certain URLs. It takes the following form:

```
CacheEnable type url
```

The *type* has to be a valid cache type as described in [Table 10.5, “Available cache types”](#). The *url* can be a path relative to the directory specified by the **DocumentRoot** directive (for example, **/images/**), a protocol (for example, **ftp://**), or an external URL such as **http://example.com/**.

Table 10.5. Available cache types

Type	Description
mem	The memory-based storage manager.
disk	The disk-based storage manager.
fd	The file descriptor cache.

Example 10.23. Using the CacheEnable directive

```
CacheEnable disk /
```


CacheLastModifiedFactor

The **CacheLastModifiedFactor** directive allows you to customize how long to cache a document that does not have any expiration date specified, but that provides information about the date of its last modification. It takes the following form:

```
CacheLastModifiedFactor number
```

The *number* is a coefficient to be used to multiply the time that passed since the last modification of the document. The default option is **0.1** (that is, one tenth).

Example 10.24. Using the CacheLastModifiedFactor directive

```
CacheLastModifiedFactor 0.1
```

CacheMaxExpire

The **CacheMaxExpire** directive allows you to specify the maximum amount of time to cache a document. It takes the following form:

```
CacheMaxExpire time
```

The *time* is specified in seconds. The default option is **86400** (that is, one day).

Example 10.25. Using the CacheMaxExpire directive

```
CacheMaxExpire 86400
```

CacheNegotiatedDocs

The **CacheNegotiatedDocs** directive allows you to enable caching of the documents that were negotiated on the basis of content. It takes the following form:

```
CacheNegotiatedDocs option
```

The *option* has to be a valid keyword as described in [Table 10.6, “Available CacheNegotiatedDocs options”](#). Since the content-negotiated documents may change over time or because of the input from the requester, the default option is **off**.

Table 10.6. Available CacheNegotiatedDocs options

Option	Description
On	Enables caching the content-negotiated documents.
off	Disables caching the content-negotiated documents.

Example 10.26. Using the CacheNegotiatedDocs directive

```
CacheNegotiatedDocs On
```

CacheRoot

The **CacheRoot** directive allows you to specify the directory to store cache files in. It takes the following form:


```
CacheRoot directory
```

The *directory* must be a full path to an existing directory in the local file system. The default option is `/var/cache/mod_proxy/`.

Example 10.27. Using the CacheRoot directive

```
CacheRoot /var/cache/mod_proxy
```

CustomLog

The **CustomLog** directive allows you to specify the log file name and the log file format. It takes the following form:

```
CustomLog path format
```

The *path* refers to a log file, and must be relative to the directory that is specified by the **ServerRoot** directive (that is, `/etc/httpd/` by default). The *format* has to be either an explicit format string, or a format name that was previously defined using the **LogFormat** directive.

Example 10.28. Using the CustomLog directive

```
CustomLog logs/access_log combined
```

DefaultIcon

The **DefaultIcon** directive allows you to specify an icon to be displayed for a file in server-generated directory listings when no other icon is associated with it. It takes the following form:

```
DefaultIcon path
```

The *path* refers to an existing icon file, and must be relative to the directory specified by the **DocumentRoot** directive (for example, `/icons/unknown.png`).

Example 10.29. Using the DefaultIcon directive

```
DefaultIcon /icons/unknown.png
```

DefaultType

The **DefaultType** directive allows you to specify a media type to be used in case the proper MIME type cannot be determined by the server. It takes the following form:

```
DefaultType content-type
```

The *content-type* has to be a valid MIME type such as **text/html**, **image/png**, **application/pdf**, etc.

Example 10.30. Using the `DefaultType` directive

```
DefaultType text/plain
```

Deny

The **Deny** directive allows you to specify which clients are denied access to a given directory. It takes the following form:

```
Deny from client...
```

The *client* can be a domain name, an IP address (both full and partial), a *network/netmask* pair, or **all** for all clients.

Example 10.31. Using the `Deny` directive

```
Deny from 192.168.1.1
```

DirectoryIndex

The **DirectoryIndex** directive allows you to specify a document to be served to a client when a directory is requested (that is, when the URL ends with the `/` character). It takes the following form:

```
DirectoryIndex filename...
```

The *filename* is a name of the file to look for in the requested directory. By default, the server looks for **index.html**, and **index.html.var**.

Example 10.32. Using the `DirectoryIndex` directive

```
DirectoryIndex index.html index.html.var
```

DocumentRoot

The **DocumentRoot** directive allows you to specify the main directory from which the content is served. It takes the following form:

```
DocumentRoot directory
```

The *directory* must be a full path to an existing directory in the local file system. The default option is **/var/www/html/**.

Example 10.33. Using the `DocumentRoot` directive

```
DocumentRoot /var/www/html
```

ErrorDocument

The **ErrorDocument** directive allows you to specify a document or a message to be displayed as a response to a particular error. It takes the following form:


```
ErrorDocument error-code action
```

The *error-code* has to be a valid code such as **403** (Forbidden), **404** (Not Found), or **500** (Internal Server Error). The *action* can be either a URL (both local and external), or a message string enclosed in double quotes (that is, ").

Example 10.34. Using the ErrorDocument directive

```
ErrorDocument 403 "Access Denied"  
ErrorDocument 404 /404-not_found.html
```

ErrorLog

The **ErrorLog** directive allows you to specify a file to which the server errors are logged. It takes the following form:

```
ErrorLog path
```

The *path* refers to a log file, and can be either absolute, or relative to the directory that is specified by the **ServerRoot** directive (that is, `/etc/httpd/` by default). The default option is **logs/error_log**

Example 10.35. Using the ErrorLog directive

```
ErrorLog logs/error_log
```

ExtendedStatus

The **ExtendedStatus** directive allows you to enable detailed server status information. It takes the following form:

```
ExtendedStatus option
```

The *option* has to be a valid keyword as described in [Table 10.7, “Available ExtendedStatus options”](#). The default option is **Off**.

Table 10.7. Available ExtendedStatus options

Option	Description
On	Enables generating the detailed server status.
Off	Disables generating the detailed server status.

Example 10.36. Using the ExtendedStatus directive

```
ExtendedStatus On
```

Group

The **Group** directive allows you to specify the group under which the `httpd` service will run. It takes the following form:


```
Group group
```

The *group* has to be an existing UNIX group. The default option is **apache**.

Note that **Group** is no longer supported inside **<VirtualHost>**, and has been replaced by the **SuexecUserGroup** directive.

Example 10.37. Using the Group directive

```
Group apache
```

HeaderName

The **HeaderName** directive allows you to specify a file to be prepended to the beginning of the server-generated directory listing. It takes the following form:

```
HeaderName filename
```

The *filename* is a name of the file to look for in the requested directory. By default, the server looks for **HEADER.html**.

Example 10.38. Using the HeaderName directive

```
HeaderName HEADER.html
```

HostnameLookups

The **HostnameLookups** directive allows you to enable automatic resolving of IP addresses. It takes the following form:

```
HostnameLookups option
```

The *option* has to be a valid keyword as described in [Table 10.8, “Available HostnameLookups options”](#). To conserve resources on the server, the default option is **Off**.

Table 10.8. Available HostnameLookups options

Option	Description
On	Enables resolving the IP address for each connection so that the hostname can be logged. However, this also adds a significant processing overhead.
Double	Enables performing the double-reverse DNS lookup. In comparison to the above option, this adds even more processing overhead.
Off	Disables resolving the IP address for each connection.

Note that when the presence of hostnames is required in server log files, it is often possible to use one of the many log analyzer tools that perform the DNS lookups more efficiently.

Example 10.39. Using the HostnameLookups directive

```
HostnameLookups Off
```


Include

The **Include** directive allows you to include other configuration files. It takes the following form:

```
Include filename
```

The **filename** can be an absolute path, a path relative to the directory specified by the **ServerRoot** directive, or a wildcard expression. All configuration files from the **/etc/httpd/conf.d/** directory are loaded by default.

Example 10.40. Using the Include directive

```
Include conf.d/*.conf
```

IndexIgnore

The **IndexIgnore** directive allows you to specify a list of file names to be omitted from the server-generated directory listings. It takes the following form:

```
IndexIgnore filename...
```

The *filename* option can be either a full file name, or a wildcard expression.

Example 10.41. Using the IndexIgnore directive

```
IndexIgnore .??* *~ *# HEADER* README* RCS CVS *,v *,t
```

IndexOptions

The **IndexOptions** directive allows you to customize the behavior of server-generated directory listings. It takes the following form:

```
IndexOptions option...
```

The *option* has to be a valid keyword as described in [Table 10.9, “Available directory listing options”](#). The default options are **Charset=UTF-8**, **FancyIndexing**, **HTMLTable**, **NameWidth=***, and **VersionSort**.

Table 10.9. Available directory listing options

Option	Description
Charset=encoding	Specifies the character set of a generated web page. The <i>encoding</i> has to be a valid character set such as UTF-8 or ISO-8859-2 .
Type=content-type	Specifies the media type of a generated web page. The <i>content-type</i> has to be a valid MIME type such as text/html or text/plain .
DescriptionWidth=value	Specifies the width of the description column. The <i>value</i> can be either a number of characters, or an asterisk (that is, *) to adjust the width automatically.

Option	Description
FancyIndexing	Enables advanced features such as different icons for certain files or possibility to re-sort a directory listing by clicking on a column header.
FolderFirst	Enables listing directories first, always placing them above files.
HTMLTable	Enables the use of HTML tables for directory listings.
IconsAreLinks	Enables using the icons as links.
IconHeight= <i>value</i>	Specifies an icon height. The <i>value</i> is a number of pixels.
IconWidth= <i>value</i>	Specifies an icon width. The <i>value</i> is a number of pixels.
IgnoreCase	Enables sorting files and directories in a case-sensitive manner.
IgnoreClient	Disables accepting query variables from a client.
NameWidth= <i>value</i>	Specifies the width of the file name column. The <i>value</i> can be either a number of characters, or an asterisk (that is, *) to adjust the width automatically.
ScanHTMLTitles	Enables parsing the file for a description (that is, the title element) in case it is not provided by the AddDescription directive.
ShowForbidden	Enables listing the files with otherwise restricted access.
SuppressColumnSorting	Disables re-sorting a directory listing by clicking on a column header.
SuppressDescription	Disables reserving a space for file descriptions.
SuppressHTMLPreamble	Disables the use of standard HTML preamble when a file specified by the HeaderName directive is present.
SuppressIcon	Disables the use of icons in directory listings.
SuppressLastModified	Disables displaying the date of the last modification field in directory listings.
SuppressRules	Disables the use of horizontal lines in directory listings.
SuppressSize	Disables displaying the file size field in directory listings.
TrackModified	Enables returning the Last-Modified and ETag values in the HTTP header.
VersionSort	Enables sorting files that contain a version number in the expected manner.
XHTML	Enables the use of XHTML 1.0 instead of the default HTML 3.2.

Example 10.42. Using the `IndexOptions` directive

```
IndexOptions FancyIndexing VersionSort NameWidth=* HTMLTable Charset=UTF-8
```

KeepAlive

The **KeepAlive** directive allows you to enable persistent connections. It takes the following form:

```
KeepAlive option
```

The *option* has to be a valid keyword as described in [Table 10.10, “Available KeepAlive options”](#). The default option is **Off**.

Table 10.10. Available KeepAlive options

Option	Description
On	Enables the persistent connections. In this case, the server will accept more than one request per connection.
Off	Disables the keep-alive connections.

Note that when the persistent connections are enabled, on a busy server, the number of child processes can increase rapidly and eventually reach the maximum limit, slowing down the server significantly. To reduce the risk, it is recommended that you set **KeepAliveTimeout** to a low number, and monitor the `/var/log/httpd/logs/error_log` log file carefully.

Example 10.43. Using the KeepAlive directive

```
KeepAlive Off
```

KeepAliveTimeout

The **KeepAliveTimeout** directive allows you to specify the amount of time to wait for another request before closing the connection. It takes the following form:

```
KeepAliveTimeout time
```

The *time* is specified in seconds. The default option is **15**.

Example 10.44. Using the KeepAliveTimeout directive

```
KeepAliveTimeout 15
```

LanguagePriority

The **LanguagePriority** directive allows you to customize the precedence of languages. It takes the following form:

```
LanguagePriority language...
```

The *language* has to be a valid MIME language such as **cs**, **en**, or **fr**.

This directive is especially useful for web servers that serve content in multiple languages based on the client's language settings.

Example 10.45. Using the LanguagePriority directive

```
LanguagePriority sk cs en
```

Listen

The *Listen* directive allows you to specify IP addresses or ports to listen to. It takes the following form:

```
Listen [ip-address:]port [protocol]
```


The *ip-address* is optional and unless supplied, the server will accept incoming requests on a given *port* from all IP addresses. Since the *protocol* is determined automatically from the port number, it can be usually omitted. The default option is to listen to port **80**.

Note that if the server is configured to listen to a port under 1024, only superuser will be able to start the `httpd` service.

Example 10.46. Using the Listen directive

```
Listen 80
```

LoadModule

The **LoadModule** directive allows you to load a *Dynamic Shared Object* (DSO) module. It takes the following form:

```
LoadModule name path
```

The *name* has to be a valid identifier of the required module. The *path* refers to an existing module file, and must be relative to the directory in which the libraries are placed (that is, **/usr/lib/httpd/** on 32-bit and **/usr/lib64/httpd/** on 64-bit systems by default).

See [Section 10.1.5, “Working with Modules”](#) for more information on the Apache HTTP Server's DSO support.

Example 10.47. Using the LoadModule directive

```
LoadModule php5_module modules/libphp5.so
```

LogFormat

The *LogFormat* directive allows you to specify a log file format. It takes the following form:

```
LogFormat format name
```

The *format* is a string consisting of options as described in [Table 10.11, “Common LogFormat options”](#). The *name* can be used instead of the format string in the **CustomLog** directive.

Table 10.11. Common LogFormat options

Option	Description
%b	Represents the size of the response in bytes.
%h	Represents the IP address or hostname of a remote client.
%l	Represents the remote log name if supplied. If not, a hyphen (that is, -) is used instead.
%r	Represents the first line of the request string as it came from the browser or client.
%s	Represents the status code.
%t	Represents the date and time of the request.
%u	If the authentication is required, it represents the remote user. If not, a hyphen (that is, -) is used instead.

Option	Description
%{field}	Represents the content of the HTTP header <i>field</i> . The common options include %{Referer} (the URL of the web page that referred the client to the server) and %{User-Agent} (the type of the web browser making the request).

Example 10.48. Using the LogFormat directive

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
```

LogLevel

The **LogLevel** directive allows you to customize the verbosity level of the error log. It takes the following form:

```
LogLevel option
```

The *option* has to be a valid keyword as described in [Table 10.12, “Available LogLevel options”](#). The default option is **warn**.

Table 10.12. Available LogLevel options

Option	Description
emerg	Only the emergency situations when the server cannot perform its work are logged.
alert	All situations when an immediate action is required are logged.
crit	All critical conditions are logged.
error	All error messages are logged.
warn	All warning messages are logged.
notice	Even normal, but still significant situations are logged.
info	Various informational messages are logged.
debug	Various debugging messages are logged.

Example 10.49. Using the LogLevel directive

```
LogLevel warn
```

MaxKeepAliveRequests

The **MaxKeepAliveRequests** directive allows you to specify the maximum number of requests for a persistent connection. It takes the following form:

```
MaxKeepAliveRequests number
```

A high *number* can improve the performance of the server. Note that using **0** allows unlimited number of requests. The default option is **100**.

Example 10.50. Using the MaxKeepAliveRequests option

```
MaxKeepAliveRequests 100
```


NameVirtualHost

The **NameVirtualHost** directive allows you to specify the IP address and port number for a name-based virtual host. It takes the following form:

```
NameVirtualHost ip-address[:port]
```

The *ip-address* can be either a full IP address, or an asterisk (that is, *****) representing all interfaces. Note that IPv6 addresses have to be enclosed in square brackets (that is, **[** and **]**). The *port* is optional.

Name-based virtual hosting allows one Apache HTTP Server to serve different domains without using multiple IP addresses.



Using secure HTTP connections

Name-based virtual hosts *only* work with non-secure HTTP connections. If using virtual hosts with a secure server, use IP address-based virtual hosts instead.

Example 10.51. Using the NameVirtualHost directive

```
NameVirtualHost *:80
```

Options

The **Options** directive allows you to specify which server features are available in a particular directory. It takes the following form:

```
Options option...
```

The *option* has to be a valid keyword as described in [Table 10.13, “Available server features”](#).

Table 10.13. Available server features

Option	Description
ExecCGI	Enables the execution of CGI scripts.
FollowSymLinks	Enables following symbolic links in the directory.
Includes	Enables server-side includes.
IncludesNOEXEC	Enables server-side includes, but does not allow the execution of commands.
Indexes	Enables server-generated directory listings.
MultiViews	Enables content-negotiated “MultiViews”.
SymLinksIfOwnerMatch	Enables following symbolic links in the directory when both the link and the target file have the same owner.
All	Enables all of the features above with the exception of MultiViews .
None	Disables all of the features above.

Example 10.52. Using the Options directive

```
Options Indexes FollowSymLinks
```

Order

The **Order** directive allows you to specify the order in which the **Allow** and **Deny** directives are evaluated. It takes the following form:

```
Order option
```

The *option* has to be a valid keyword as described in [Table 10.14, “Available Order options”](#). The default option is **allow,deny**.

Table 10.14. Available Order options

Option	Description
allow,deny	Allow directives are evaluated first.
deny,allow	Deny directives are evaluated first.

Example 10.53. Using the Order directive

```
Order allow,deny
```

PidFile

The **PidFile** directive allows you to specify a file to which the *process ID* (PID) of the server is stored. It takes the following form:

```
PidFile path
```

The *path* refers to a pid file, and can be either absolute, or relative to the directory that is specified by the **ServerRoot** directive (that is, **/etc/httpd/** by default). The default option is **run/httpd.pid**.

Example 10.54. Using the PidFile directive

```
PidFile run/httpd.pid
```

ProxyRequests

The **ProxyRequests** directive allows you to enable forward proxy requests. It takes the following form:

```
ProxyRequests option
```

The *option* has to be a valid keyword as described in [Table 10.15, “Available ProxyRequests options”](#). The default option is **Off**.

Table 10.15. Available ProxyRequests options

Option	Description
On	Enables forward proxy requests.
Off	Disables forward proxy requests.

Example 10.55. Using the ProxyRequests directive

```
ProxyRequests On
```

ReadmeName

The **ReadmeName** directive allows you to specify a file to be appended to the end of the server-generated directory listing. It takes the following form:

```
ReadmeName filename
```

The *filename* is a name of the file to look for in the requested directory. By default, the server looks for **README.html**.

Example 10.56. Using the ReadmeName directive

```
ReadmeName README.html
```

Redirect

The **Redirect** directive allows you to redirect a client to another URL. It takes the following form:

```
Redirect [status] path url
```

The *status* is optional, and if provided, it has to be a valid keyword as described in [Table 10.16, “Available status options”](#). The *path* refers to the old location, and must be relative to the directory specified by the **DocumentRoot** directive (for example, **/docs**). The *url* refers to the current location of the content (for example, **http://docs.example.com**).

Table 10.16. Available status options

Status	Description
permanent	Indicates that the requested resource has been moved permanently. The 301 (Moved Permanently) status code is returned to a client.
temp	Indicates that the requested resource has been moved only temporarily. The 302 (Found) status code is returned to a client.
seeother	Indicates that the requested resource has been replaced. The 303 (See Other) status code is returned to a client.
gone	Indicates that the requested resource has been removed permanently. The 410 (Gone) status is returned to a client.

Note that for more advanced redirection techniques, you can use the `mod_rewrite` module that is part of the Apache HTTP Server installation.

Example 10.57. Using the Redirect directive

```
Redirect permanent /docs http://docs.example.com
```

ScriptAlias

The **ScriptAlias** directive allows you to specify the location of CGI scripts. It takes the following form:

```
ScriptAlias url-path real-path
```

The *url-path* must be relative to the directory specified by the **DocumentRoot** directive (for example, **/cgi-bin/**). The *real-path* is a full path to a file or directory in the local file system.

This directive is typically followed by the **Directory** tag with additional permissions to access the target directory. By default, the **/cgi-bin/** alias is created so that the scripts located in the **/var/www/cgi-bin/** are accessible.

The **ScriptAlias** directive is used for security reasons to prevent CGI scripts from being viewed as ordinary text documents.

Example 10.58. Using the ScriptAlias directive

```
ScriptAlias /cgi-bin/ /var/www/cgi-bin/

<Directory "/var/www/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>
```

ServerAdmin

The **ServerAdmin** directive allows you to specify the email address of the server administrator to be displayed in server-generated web pages. It takes the following form:

```
ServerAdmin email
```

The default option is **root@localhost**.

This directive is commonly set to **webmaster@hostname**, where *hostname* is the address of the server. Once set, alias **webmaster** to the person responsible for the web server in **/etc/aliases**, and as superuser, run the **newaliases** command.

Example 10.59. Using the ServerAdmin directive

```
ServerAdmin webmaster@penguin.example.com
```

ServerName

The **ServerName** directive allows you to specify the hostname and the port number of a web server. It takes the following form:


```
ServerName hostname[:port]
```

The *hostname* has to be a *fully qualified domain name* (FQDN) of the server. The *port* is optional, but when supplied, it has to match the number specified by the **Listen** directive.

When using this directive, make sure that the IP address and server name pair are included in the **/etc/hosts** file.

Example 10.60. Using the ServerName directive

```
ServerName penguin.example.com:80
```

ServerRoot

The **ServerRoot** directive allows you to specify the directory in which the server operates. It takes the following form:

```
ServerRoot directory
```

The *directory* must be a full path to an existing directory in the local file system. The default option is **/etc/httpd/**.

Example 10.61. Using the ServerRoot directive

```
ServerRoot /etc/httpd
```

ServerSignature

The **ServerSignature** directive allows you to enable displaying information about the server on server-generated documents. It takes the following form:

```
ServerSignature option
```

The *option* has to be a valid keyword as described in [Table 10.17, “Available ServerSignature options”](#). The default option is **On**.

Table 10.17. Available ServerSignature options

Option	Description
On	Enables appending the server name and version to server-generated pages.
Off	Disables appending the server name and version to server-generated pages.
EMail	Enables appending the server name, version, and the email address of the system administrator as specified by the ServerAdmin directive to server-generated pages.

Example 10.62. Using the ServerSignature directive

```
ServerSignature On
```


ServerTokens

The **ServerTokens** directive allows you to customize what information are included in the Server response header. It takes the following form:

```
ServerTokens option
```

The *option* has to be a valid keyword as described in [Table 10.18, “Available ServerTokens options”](#). The default option is **OS**.

Table 10.18. Available ServerTokens options

Option	Description
Prod	Includes the product name only (that is, Apache).
Major	Includes the product name and the major version of the server (for example, 2).
Minor	Includes the product name and the minor version of the server (for example, 2.2).
Min	Includes the product name and the minimal version of the server (for example, 2.2.15).
OS	Includes the product name, the minimal version of the server, and the type of the operating system it is running on (for example, Red Hat).
Full	Includes all the information above along with the list of loaded modules.

Note that for security reasons, it is recommended to reveal as little information about the server as possible.

Example 10.63. Using the ServerTokens directive

```
ServerTokens Prod
```

SuexecUserGroup

The **SuexecUserGroup** directive allows you to specify the user and group under which the CGI scripts will be run. It takes the following form:

```
SuexecUserGroup user group
```

The *user* has to be an existing user, and the *group* must be a valid UNIX group.

For security reasons, the CGI scripts should not be run with root privileges. Note that in **<VirtualHost>**, **SuexecUserGroup** replaces the **User** and **Group** directives.

Example 10.64. Using the SuexecUserGroup directive

```
SuexecUserGroup apache apache
```

Timeout

The **Timeout** directive allows you to specify the amount of time to wait for an event before closing a connection. It takes the following form:

```
Timeout time
```


The *time* is specified in seconds. The default option is **60**.

Example 10.65. Using the Timeout directive

```
Timeout 60
```

TypesConfig

The **TypesConfig** allows you to specify the location of the MIME types configuration file. It takes the following form:

```
TypesConfig path
```

The *path* refers to an existing MIME types configuration file, and can be either absolute, or relative to the directory that is specified by the **ServerRoot** directive (that is, **/etc/httpd/** by default). The default option is **/etc/mime.types**.

Note that instead of editing **/etc/mime.types**, the recommended way to add MIME type mapping to the Apache HTTP Server is to use the **AddType** directive.

Example 10.66. Using the TypesConfig directive

```
TypesConfig /etc/mime.types
```

UseCanonicalName

The **UseCanonicalName** allows you to specify the way the server refers to itself. It takes the following form:

```
UseCanonicalName option
```

The *option* has to be a valid keyword as described in [Table 10.19, “Available UseCanonicalName options”](#). The default option is **Off**.

Table 10.19. Available UseCanonicalName options

Option	Description
On	Enables the use of the name that is specified by the ServerName directive.
Off	Disables the use of the name that is specified by the ServerName directive. The hostname and port number provided by the requesting client are used instead.
DNS	Disables the use of the name that is specified by the ServerName directive. The hostname determined by a reverse DNS lookup is used instead.

Example 10.67. Using the UseCanonicalName directive

```
UseCanonicalName Off
```


User

The **User** directive allows you to specify the user under which the `httpd` service will run. It takes the following form:

```
User user
```

The *user* has to be an existing UNIX user. The default option is **apache**.

For security reasons, the `httpd` service should not be run with `root` privileges. Note that **User** is no longer supported inside **<VirtualHost>**, and has been replaced by the **SuexecUserGroup** directive.

Example 10.68. Using the User directive

```
User apache
```

UserDir

The **UserDir** directive allows you to enable serving content from users' home directories. It takes the following form:

```
UserDir option
```

The *option* can be either a name of the directory to look for in user's home directory (typically **public_html**), or a valid keyword as described in [Table 10.20, "Available UserDir options"](#). The default option is **disabled**.

Table 10.20. Available UserDir options

Option	Description
enabled <i>user...</i>	Enables serving content from home directories of given <i>users</i> .
disabled [<i>user...</i>]	Disables serving content from home directories, either for all users, or, if a space separated list of <i>users</i> is supplied, for given users only.

Set the correct permissions

In order for the web server to access the content, the permissions on relevant directories and files must be set correctly. Make sure that all users are able to access the home directories, and that they can access and read the content of the directory specified by the **UserDir** directive. For example, to allow access to **public_html/** in the home directory of user `joe`, type the following at a shell prompt as `root`:

```
~]# chmod a+x /home/joe/
~]# chmod a+rx /home/joe/public_html/
```

All files in this directory must be set accordingly.

Example 10.69. Using the UserDir directive

```
UserDir public_html
```

10.1.4.2. Common ssl.conf Directives

The *Secure Sockets Layer* (SSL) directives allow you to customize the behavior of the Apache HTTP Secure Server, and in most cases, they are configured appropriately during the installation. Be careful when changing these settings, as incorrect configuration can lead to security vulnerabilities.

The following directive is commonly used in `/etc/httpd/conf.d/ssl.conf`:

SetEnvIf

The **SetEnvIf** directive allows you to set environment variables based on the headers of incoming connections. It takes the following form:

```
SetEnvIf option pattern [!]variable[=value]...
```

The *option* can be either a HTTP header field, a previously defined environment variable name, or a valid keyword as described in [Table 10.21, “Available SetEnvIf options”](#). The *pattern* is a regular expression. The *variable* is an environment variable that is set when the option matches the pattern. If the optional exclamation mark (that is, **!**) is present, the variable is removed instead of being set.

Table 10.21. Available SetEnvIf options

Option	Description
Remote_Host	Refers to the client's hostname.
Remote_Addr	Refers to the client's IP address.
Server_Addr	Refers to the server's IP address.
Request_Method	Refers to the request method (for example, GET).
Request_Protocol	Refers to the protocol name and version (for example, HTTP/1.1).
Request_URI	Refers to the requested resource.

The **SetEnvIf** directive is used to disable HTTP keepalives, and to allow SSL to close the connection without a closing notification from the client browser. This is necessary for certain web browsers that do not reliably shut down the SSL connection.

Example 10.70. Using the SetEnvIf directive

```
SetEnvIf User-Agent ".*MSIE.*" \
    nokeepalive ssl-unclean-shutdown \
    downgrade-1.0 force-response-1.0
```

Note that for the `/etc/httpd/conf.d/ssl.conf` file to be present, the `mod_ssl` needs to be installed. See [Section 10.1.7, “Setting Up an SSL Server”](#) for more information on how to install and configure an SSL server.

10.1.4.3. Common Multi-Processing Module Directives

The *Multi-Processing Module* (MPM) directives allow you to customize the behavior of a particular MPM specific server-pool. Since its characteristics differ depending on which MPM is used, the directives are embedded in **IfModule**. By default, the server-pool is defined for both the prefork and worker MPMs.

The following MPM directives are commonly used in `/etc/httpd/conf/httpd.conf`:

MaxClients

The **MaxClients** directive allows you to specify the maximum number of simultaneously connected clients to process at one time. It takes the following form:

```
MaxClients number
```

A high *number* can improve the performance of the server, although it is not recommended to exceed **256** when using the prefork MPM.

Example 10.71. Using the MaxClients directive

```
MaxClients 256
```

MaxRequestsPerChild

The **MaxRequestsPerChild** directive allows you to specify the maximum number of request a child process can serve before it dies. It takes the following form:

```
MaxRequestsPerChild number
```

Setting the *number* to **0** allows unlimited number of requests.

The **MaxRequestsPerChild** directive is used to prevent long-lived processes from causing memory leaks.

Example 10.72. Using the MaxRequestsPerChild directive

```
MaxRequestsPerChild 4000
```

MaxSpareServers

The **MaxSpareServers** directive allows you to specify the maximum number of spare child processes. It takes the following form:

```
MaxSpareServers number
```

This directive is used by the prefork MPM only.

Example 10.73. Using the MaxSpareServers directive

```
MaxSpareServers 20
```


MaxSpareThreads

The **MaxSpareThreads** directive allows you to specify the maximum number of spare server threads. It takes the following form:

```
MaxSpareThreads number
```

The *number* must be greater than or equal to the sum of **MinSpareThreads** and **ThreadsPerChild**. This directive is used by the worker MPM only.

Example 10.74. Using the MaxSpareThreads directive

```
MaxSpareThreads 75
```

MinSpareServers

The **MinSpareServers** directive allows you to specify the minimum number of spare child processes. It takes the following form:

```
MinSpareServers number
```

Note that a high *number* can create a heavy processing load on the server. This directive is used by the prefork MPM only.

Example 10.75. Using the MinSpareServers directive

```
MinSpareServers 5
```

MinSpareThreads

The **MinSpareThreads** directive allows you to specify the minimum number of spare server threads. It takes the following form:

```
MinSpareThreads number
```

This directive is used by the worker MPM only.

Example 10.76. Using the MinSpareThreads directive

```
MinSpareThreads 75
```

StartServers

The **StartServers** directive allows you to specify the number of child processes to create when the service is started. It takes the following form:

```
StartServers number
```

Since the child processes are dynamically created and terminated according to the current traffic load, it is usually not necessary to change this value.

Example 10.77. Using the StartServers directive

```
StartServers 8
```

ThreadsPerChild

The **ThreadsPerChild** directive allows you to specify the number of threads a child process can create. It takes the following form:

```
ThreadsPerChild number
```

This directive is used by the worker MPM only.

Example 10.78. Using the ThreadsPerChild directive

```
ThreadsPerChild 25
```

10.1.5. Working with Modules

Being a modular application, the `httpd` service is distributed along with a number of *Dynamic Shared Objects* (DSOs), which can be dynamically loaded or unloaded at runtime as necessary. By default, these modules are located in `/usr/lib/httpd/modules/` on 32-bit and in `/usr/lib64/httpd/modules/` on 64-bit systems.

10.1.5.1. Loading a Module

To load a particular DSO module, use the **LoadModule** directive as described in [Section 10.1.4.1, “Common httpd.conf Directives”](#). Note that modules provided by a separate package often have their own configuration file in the `/etc/httpd/conf.d/` directory.

Example 10.79. Loading the mod_ssl DSO

```
LoadModule ssl_module modules/mod_ssl.so
```

Once you are finished, restart the web server to reload the configuration. See [Section 10.1.3.3, “Restarting the Service”](#) for more information on how to restart the `httpd` service.

10.1.5.2. Writing a Module

If you intend to create a new DSO module, make sure you have the `httpd-devel` package installed. To do so, enter the following command as root:

```
~]# dnf install httpd-devel
```

This package contains the include files, the header files, and the **APache eXtenSion (apxs)** utility required to compile a module.

Once written, you can build the module with the following command:


```
~]# apxs -i -a -c module_name.c
```

If the build was successful, you should be able to load the module the same way as any other module that is distributed with the Apache HTTP Server.

10.1.6. Setting Up Virtual Hosts

The Apache HTTP Server's built in virtual hosting allows the server to provide different information based on which IP address, host name, or port is being requested.

To create a name-based virtual host, copy the example configuration file `/usr/share/doc/httpd-VERSION/httpd-vhosts.conf` into the `/etc/httpd/conf.d/` directory, and replace the `@@Port@@` and `@@ServerRoot@@` placeholder values. Customize the options according to your requirements as shown in [Example 10.80, "Example virtual host configuration"](#).

Example 10.80. Example virtual host configuration

```
<VirtualHost *:80>
    ServerAdmin webmaster@penguin.example.com
    DocumentRoot "/www/docs/penguin.example.com"
    ServerName penguin.example.com
    ServerAlias www.penguin.example.com
    ErrorLog "/var/log/httpd/dummy-host.example.com-error_log"
    CustomLog "/var/log/httpd/dummy-host.example.com-access_log" common
</VirtualHost>
```

Note that **ServerName** must be a valid DNS name assigned to the machine. The **<VirtualHost>** container is highly customizable, and accepts most of the directives available within the main server configuration. Directives that are *not* supported within this container include **User** and **Group**, which were replaced by **SuexecUserGroup**.



Changing the port number

If you configure a virtual host to listen on a non-default port, make sure you update the **Listen** directive in the global settings section of the `/etc/httpd/conf/httpd.conf` file accordingly.

To activate a newly created virtual host, the web server has to be restarted first. See [Section 10.1.3.3, "Restarting the Service"](#) for more information on how to restart the `httpd` service.

10.1.7. Setting Up an SSL Server

Secure Sockets Layer (SSL) is a cryptographic protocol that allows a server and a client to communicate securely. Along with its extended and improved version called *Transport Layer Security* (TLS), it ensures both privacy and data integrity. The Apache HTTP Server in combination with `mod_ssl`, a module that uses the OpenSSL toolkit to provide the SSL/TLS support, is commonly referred to as the *SSL server*.

Unlike a regular HTTP connection that can be read and possibly modified by anybody who is able to intercept it, the use of `mod_ssl` prevents any inspection or modification of the transmitted content.

This section provides basic information on how to enable this module in the Apache HTTP Server configuration, and guides you through the process of generating private keys and self-signed certificates.

10.1.7.1. An Overview of Certificates and Security

Secure communication is based on the use of keys. In conventional or *symmetric cryptography*, both ends of the transaction have the same key they can use to decode each other's transmissions. On the other hand, in public or *asymmetric cryptography*, two keys co-exist: a *private key* that is kept a secret, and a *public key* that is usually shared with the public. While the data encoded with the public key can only be decoded with the private key, data encoded with the private key can in turn only be decoded with the public key.

To provide secure communications using SSL, an SSL server must use a digital certificate signed by a *Certificate Authority* (CA). The certificate lists various attributes of the server (that is, the server host name, the name of the company, its location, etc.), and the signature produced using the CA's private key. This signature ensures that a particular certificate authority has signed the certificate, and that the certificate has not been modified in any way.

When a web browser establishes a new SSL connection, it checks the certificate provided by the web server. If the certificate does not have a signature from a trusted CA, or if the host name listed in the certificate does not match the host name used to establish the connection, it refuses to communicate with the server and usually presents a user with an appropriate error message.

By default, most web browsers are configured to trust a set of widely used certificate authorities. Because of this, an appropriate CA should be chosen when setting up a secure server, so that target users can trust the connection, otherwise they will be presented with an error message, and will have to accept the certificate manually. Since encouraging users to override certificate errors can allow an attacker to intercept the connection, you should use a trusted CA whenever possible. For more information on this, see [Table 10.22, "Information about CA lists used by common web browsers"](#).

Table 10.22. Information about CA lists used by common web browsers

Web Browser	Link
Mozilla Firefox	Mozilla root CA list² .
Opera	Information on root certificates used by Opera³ .
Internet Explorer	Information on root certificates used by Microsoft Windows⁴ .
Chromium	Information on root certificates used by the Chromium project⁵ .

When setting up an SSL server, you need to generate a certificate request and a private key, and then send the certificate request, proof of the company's identity, and payment to a certificate authority. Once the CA verifies the certificate request and your identity, it will send you a signed certificate you can use with your server. Alternatively, you can create a self-signed certificate that does not contain a CA signature, and thus should be used for testing purposes only.

² <http://www.mozilla.org/projects/security/certs/included/>

³ <http://www.opera.com/docs/ca/>

⁴ <http://support.microsoft.com/kb/931125>

⁵ <http://www.chromium.org/Home/chromium-security/root-ca-policy>

10.1.7.2. Enabling the mod_ssl Module

If you intend to set up an SSL server, make sure you have the *mod_ssl* (the *mod_ssl* module) and *openssl* (the OpenSSL toolkit) packages installed. To do so, enter the following command as root:

```
~]# dnf install mod_ssl openssl
```

This will create the *mod_ssl* configuration file at `/etc/httpd/conf.d/ssl.conf`, which is included in the main Apache HTTP Server configuration file by default. For the module to be loaded, restart the *httpd* service as described in [Section 10.1.3.3, “Restarting the Service”](#).



Important

Due to the SSL3.0 protocol vulnerability CVE-2014-3566, described in [SSL 3.0 Protocol Vulnerability and POODLE Attack](#)⁶, it is recommended to disable SSL and use only TLSv1.1 or TLSv1.2. Backwards compatibility can be achieved using TLSv1.0. Many products have the ability to use SSLv2 or SSLv3 protocols, or enable them by default. However, the use of SSLv2 or SSLv3 is now strongly recommended against.

10.1.7.3. Enabling and Disabling SSL and TLS in mod_ssl

To disable and enable specific versions of the SSL and TLS protocol, either do it globally by adding the **SSLProtocol** directive in the “## SSL Global Context” section of the configuration file and removing it everywhere else, or edit the default entry under “# SSL Protocol support” in all “VirtualHost” sections. If you do not specify it in the per-domain VirtualHost section then it will inherit the settings from the global section. To make sure that a protocol version is being disabled the administrator should either **only** specify **SSLProtocol** in the “SSL Global Context” section, or specify it in **all** per-domain VirtualHost sections.

Procedure 10.1. Disable SSLv2 and SSLv3

To disable SSL version 2 and SSL version 3, which implies enabling everything except SSL version 2 and SSL version 3, in all VirtualHost sections, proceed as follows:

1. As root, open the `/etc/httpd/conf.d/ssl.conf` file and search for **all** instances of the **SSLProtocol** directive. By default, the configuration file contains one section that looks as follows:

```
~]# vi /etc/httpd/conf.d/ssl.conf
# SSL Protocol support:
# List the enable protocol levels with which clients will be able to
# connect.  Disable SSLv2 access by default:
SSLProtocol all -SSLv2
```

This section is within the VirtualHost section.

2. Edit the **SSLProtocol** line as follows:

```
# SSL Protocol support:
# List the enable protocol levels with which clients will be able to
```

⁶ <https://www.us-cert.gov/ncas/alerts/TA14-290A>


```
# connect.  Disable SSLv2 access by default:
SSLProtocol All -SSLv2 -SSLv3
```

Repeat this action for all VirtualHost sections.

3. Verify that all occurrences of the **SSLProtocol** directive have been changed as follows:

```
~]# grep SSLProtocol /etc/httpd/conf.d/ssl.conf
SSLProtocol all -SSLv2 -SSLv3
```

This step is particularly important if you have more than the one default VirtualHost section.

4. Restart the Apache daemon as follows:

```
~]# service httpd restart
```

Note that any sessions will be interrupted.

10.1.7.4. Using an Existing Key and Certificate

If you have a previously created key and certificate, you can configure the SSL server to use these files instead of generating new ones. There are only two situations where this is not possible:

1. *You are changing the IP address or domain name.*

Certificates are issued for a particular IP address and domain name pair. If one of these values changes, the certificate becomes invalid.

2. *You have a certificate from VeriSign, and you are changing the server software.*

VeriSign, a widely used certificate authority, issues certificates for a particular software product, IP address, and domain name. Changing the software product renders the certificate invalid.

In either of the above cases, you will need to obtain a new certificate. For more information on this topic, see [Section 10.1.7.5, “Generating a New Key and Certificate”](#).

If you want to use an existing key and certificate, move the relevant files to the **/etc/pki/tls/private/** and **/etc/pki/tls/certs/** directories respectively. You can do so by issuing the following commands as root:

```
~]# mv key_file.key /etc/pki/tls/private/hostname.key
~]# mv certificate.crt /etc/pki/tls/certs/hostname.crt
```

Then add the following lines to the **/etc/httpd/conf.d/ssl.conf** configuration file:

```
SSLCertificateFile /etc/pki/tls/certs/hostname.crt
SSLCertificateKeyFile /etc/pki/tls/private/hostname.key
```

To load the updated configuration, restart the `httpd` service as described in [Section 10.1.3.3, “Restarting the Service”](#).

Example 10.81. Using a key and certificate from the Red Hat Secure Web Server

```
~]# mv /etc/httpd/conf/httpsd.key /etc/pki/tls/private/penguin.example.com.key
```



```
~]# mv /etc/httpd/conf/httpsd.crt /etc/pki/tls/certs/penguin.example.com.crt
```

10.1.7.5. Generating a New Key and Certificate

In order to generate a new key and certificate pair, the *crypto-utils* package must be installed on the system. To install it, enter the following command as root:

```
~]# dnf install crypto-utils
```

This package provides a set of tools to generate and manage SSL certificates and private keys, and includes **genkey**, the Red Hat Keypair Generation utility that will guide you through the key generation process.



Replacing an existing certificate

If the server already has a valid certificate and you are replacing it with a new one, specify a different serial number. This ensures that client browsers are notified of this change, update to this new certificate as expected, and do not fail to access the page. To create a new certificate with a custom serial number, use the following command instead of **genkey**:

```
~]# openssl req -x509 -new -set_serial number -key hostname.key -out hostname.crt
```



Remove a previously created key

If there already is a key file for a particular host name in your system, **genkey** will refuse to start. In this case, remove the existing file using the following command as root:

```
~]# rm /etc/pki/tls/private/hostname.key
```

To run the utility enter the **genkey** command as root, followed by the appropriate host name (for example, *penguin.example.com*):

```
~]# genkey hostname
```

To complete the key and certificate creation, take the following steps:

1. Review the target locations in which the key and certificate will be stored.

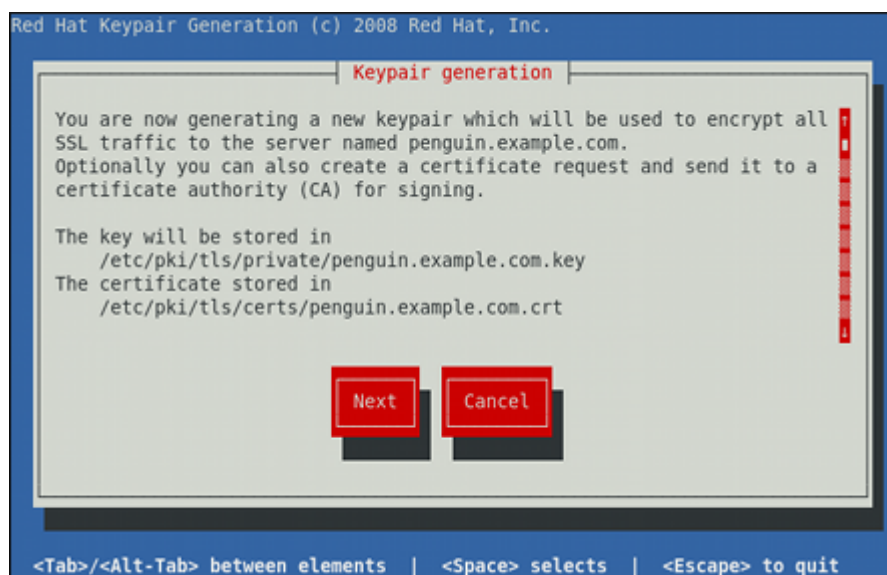


Figure 10.1. Running the genkey utility

Use the **Tab** key to select the **Next** button, and press **Enter** to proceed to the next screen.

2. Using the **up** and **down** arrow keys, select a suitable key size. Note that while a larger key increases the security, it also increases the response time of your server. The NIST recommends using **2048 bits**. See [NIST Special Publication 800-131A](http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf)⁷.

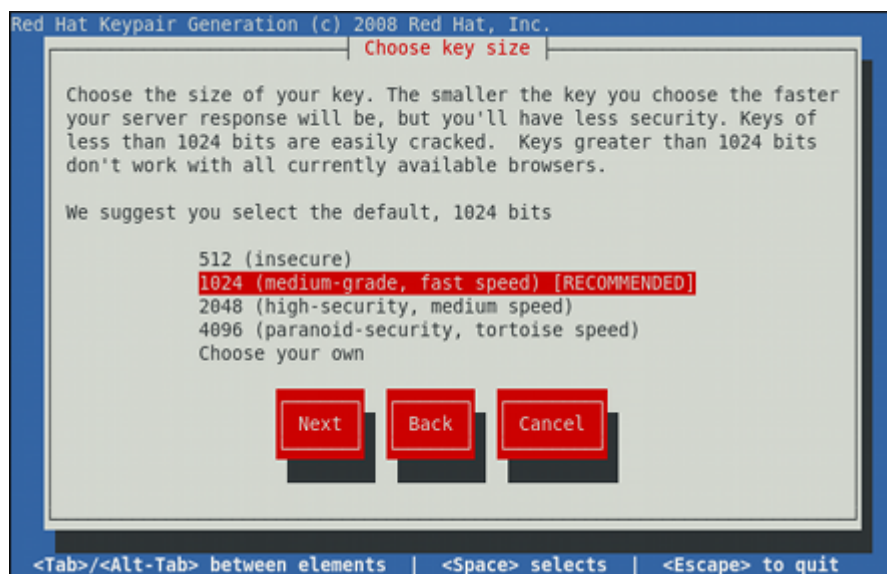


Figure 10.2. Selecting the key size

Once finished, use the **Tab** key to select the **Next** button, and press **Enter** to initiate the random bits generation process. Depending on the selected key size, this may take some time.

⁷ <http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf>

3. Decide whether you want to send a certificate request to a certificate authority.

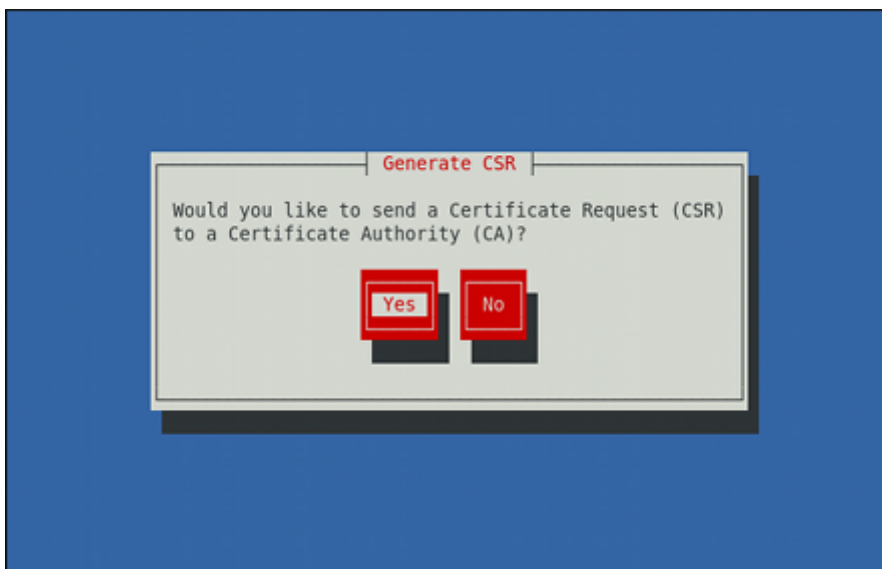


Figure 10.3. Generating a certificate request

Use the **Tab** key to select **Yes** to compose a certificate request, or **No** to generate a self-signed certificate. Then press **Enter** to confirm your choice.

4. Using the **Spacebar** key, enable (**[*]**) or disable (**[]**) the encryption of the private key.



Figure 10.4. Encrypting the private key

Use the **Tab** key to select the **Next** button, and press **Enter** to proceed to the next screen.

5. If you have enabled the private key encryption, enter an adequate passphrase. Note that for security reasons, it is not displayed as you type, and it must be at least five characters long.



Figure 10.5. Entering a passphrase

Use the **Tab** key to select the **Next** button, and press **Enter** to proceed to the next screen.



Do not forget the passphrase

Entering the correct passphrase is required in order for the server to start. If you lose it, you will need to generate a new key and certificate.

6. Customize the certificate details.



Figure 10.6. Specifying certificate information

Use the **Tab** key to select the **Next** button, and press **Enter** to finish the key generation.

7. If you have previously enabled the certificate request generation, you will be prompted to send it to a certificate authority.

```

You now need to submit your CSR and documentation to your certificate
authority. Submitting your CSR may involve pasting it into an online
web form, or mailing it to a specific address. In either case, you
should include the BEGIN and END lines.

-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBqjCCARMCAQAwajELMAkGA1UEBhMCR0IxIjAQBgNVBAGTCUJlcmtzaGlyZTEQ
MA4GA1UEBxMHTmV3YnVyeTEXMBUGA1UEChMOTXkgQ29tcGFueSBMdGQxHDAaBgNV
BAMTE3Blbmd1aW4uZXhhbXBsZS5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJ
AoGBAJjw8bXq7WKGXNZsNZltEe9849wUMc4uAh+X8251b8x+ptJQCanGeNhLLXU
xil5srY2TjotSQ5DvyFgPQmFFe3cn7v//bKNgNqd4h0EbrFGaj/hDUG3fXnjujkX
hP+9iY/eIAQZlHQSkABh/2egtIllpfDeRvsTUX376TnkIWLhAgMBAAAGgADANBgkq
hkiG9w0BAQQAQFAA0BgQBUTjgjcnts1hZK070c5j+b4IfsBCwm4lnvGx3j0wpLdRq/
rHpx5cbHV99vcKnF3CwDrze9DgpTdjdbAccSCVgSG5G6E8JZXWYD8EK8p2naJNQL1
YVX1KPisMPLZuZ9cTb+k4K0cbug0IQiYaKNLNI/0zLE1VEWZYFX0UBFM2gXYw==
-----END NEW CERTIFICATE REQUEST-----

A copy of this CSR has been saved in the file
/etc/pki/tls/certs/penguin.example.com.1.csr

Press return when ready to continue

```

Figure 10.7. Instructions on how to send a certificate request

Press **Enter** to return to a shell prompt.

Once generated, add the key and certificate locations to the `/etc/httpd/conf.d/ssl.conf` configuration file:

```

SSLCertificateFile /etc/pki/tls/certs/hostname.crt
SSLCertificateKeyFile /etc/pki/tls/private/hostname.key

```

Finally, restart the `httpd` service as described in [Section 10.1.3.3, “Restarting the Service”](#), so that the updated configuration is loaded.

10.1.8. Additional Resources

To learn more about the Apache HTTP Server, see the following resources.

Installed Documentation

- **httpd(8)** — The manual page for the `httpd` service containing the complete list of its command-line options.
- **genkey(1)** — The manual page for **genkey** utility, provided by the *crypto-utils* package.
- **apachectl(8)** — The manual page for the Apache HTTP Server Control Interface.

Installable Documentation

- <http://localhost/manual/> — The official documentation for the Apache HTTP Server with the full description of its directives and available modules. Note that in order to access this documentation, you must have the *httpd-manual* package installed, and the web server must be running.

Before accessing the documentation, issue the following commands as root:


```
~]# dnf install httpd-manual  
~]# apachectl graceful
```

Online Documentation

- <http://httpd.apache.org/> — The official website for the Apache HTTP Server with documentation on all the directives and default modules.
- `ulink url="http://www.modssl.org/" />` — The official website for the **mod_ssl** module.
- <http://www.openssl.org/> — The OpenSSL home page containing further documentation, frequently asked questions, links to the mailing lists, and other useful resources.

Mail Servers

Fedora offers many advanced applications to serve and access email. This chapter describes modern email protocols in use today, and some of the programs designed to send and receive email.

11.1. Email Protocols

Today, email is delivered using a client/server architecture. An email message is created using a mail client program. This program then sends the message to a server. The server then forwards the message to the recipient's email server, where the message is then supplied to the recipient's email client.

To enable this process, a variety of standard network protocols allow different machines, often running different operating systems and using different email programs, to send and receive email.

The following protocols discussed are the most commonly used in the transfer of email.

11.1.1. Mail Transport Protocols

Mail delivery from a client application to the server, and from an originating server to the destination server, is handled by the *Simple Mail Transfer Protocol (SMTP)*.

11.1.1.1. SMTP

The primary purpose of SMTP is to transfer email between mail servers. However, it is critical for email clients as well. To send email, the client sends the message to an outgoing mail server, which in turn contacts the destination mail server for delivery. For this reason, it is necessary to specify an SMTP server when configuring an email client.

Under Fedora, a user can configure an SMTP server on the local machine to handle mail delivery. However, it is also possible to configure remote SMTP servers for outgoing mail.

One important point to make about the SMTP protocol is that it does not require authentication. This allows anyone on the Internet to send email to anyone else or even to large groups of people. It is this characteristic of SMTP that makes junk email or *spam* possible. Imposing relay restrictions limits random users on the Internet from sending email through your SMTP server, to other servers on the internet. Servers that do not impose such restrictions are called *open relay* servers.

Fedora provides the Postfix and Sendmail SMTP programs.

11.1.2. Mail Access Protocols

There are two primary protocols used by email client applications to retrieve email from mail servers: the *Post Office Protocol (POP)* and the *Internet Message Access Protocol (IMAP)*.

11.1.2.1. POP

The default POP server under Fedora is **Dovecot** and is provided by the *dovecot* package.



Installing the dovecot package

In order to use **Dovecot**, first ensure the *dovecot* package is installed on your system by running, as root:

```
~]# dnf install dovecot
```

For more information on installing packages with DNF, see [Section 6.2.4, “Installing Packages”](#).

When using a POP server, email messages are downloaded by email client applications. By default, most POP email clients are automatically configured to delete the message on the email server after it has been successfully transferred, however this setting usually can be changed.

POP is fully compatible with important Internet messaging standards, such as *Multipurpose Internet Mail Extensions (MIME)*, which allow for email attachments.

POP works best for users who have one system on which to read email. It also works well for users who do not have a persistent connection to the Internet or the network containing the mail server. Unfortunately for those with slow network connections, POP requires client programs upon authentication to download the entire content of each message. This can take a long time if any messages have large attachments.

The most current version of the standard POP protocol is POP3.

There are, however, a variety of lesser-used POP protocol variants:

- *APOP* — POP3 with MD5 authentication. An encoded hash of the user's password is sent from the email client to the server rather than sending an unencrypted password.
- *KPOP* — POP3 with Kerberos authentication.
- *RPOP* — POP3 with RPOP authentication. This uses a per-user ID, similar to a password, to authenticate POP requests. However, this ID is not encrypted, so RPOP is no more secure than standard POP.

For added security, it is possible to use *Secure Socket Layer (SSL)* encryption for client authentication and data transfer sessions. This can be enabled by using the **pop3s** service, or by using the **stunnel** application. For more information on securing email communication, see [Section 11.5.1, “Securing Communication”](#).

11.1.2.2. IMAP

The default IMAP server under Fedora is **Dovecot** and is provided by the *dovecot* package. See [Section 11.1.2.1, “POP”](#) for information on how to install **Dovecot**.

When using an IMAP mail server, email messages remain on the server where users can read or delete them. IMAP also allows client applications to create, rename, or delete mail directories on the server to organize and store email.

IMAP is particularly useful for users who access their email using multiple machines. The protocol is also convenient for users connecting to the mail server via a slow connection, because only the email

header information is downloaded for messages until opened, saving bandwidth. The user also has the ability to delete messages without viewing or downloading them.

For convenience, IMAP client applications are capable of caching copies of messages locally, so the user can browse previously read messages when not directly connected to the IMAP server.

IMAP, like POP, is fully compatible with important Internet messaging standards, such as MIME, which allow for email attachments.

For added security, it is possible to use SSL encryption for client authentication and data transfer sessions. This can be enabled by using the **imaps** service, or by using the **stunnel** program. For more information on securing email communication, see [Section 11.5.1, “Securing Communication”](#).

Other free, as well as commercial, IMAP clients and servers are available, many of which extend the IMAP protocol and provide additional functionality.

11.1.2.3. Dovecot

The **imap-login** and **pop3-login** processes which implement the IMAP and POP3 protocols are spawned by the master **dovecot** daemon included in the *dovecot* package. The use of IMAP and POP is configured through the **/etc/dovecot/dovecot.conf** configuration file; by default **dovecot** runs IMAP and POP3 together with their secure versions using SSL. To configure **dovecot** to use POP, complete the following steps:

1. Edit the **/etc/dovecot/dovecot.conf** configuration file to make sure the **protocols** variable is uncommented (remove the hash sign (#) at the beginning of the line) and contains the **pop3** argument. For example:

```
protocols = imap pop3 lmtp
```

When the **protocols** variable is left commented out, **dovecot** will use the default values as described above.

2. Make the change operational for the current session by running the following command as root:

```
~]# systemctl restart dovecot
```

3. Make the change operational after the next reboot by running the command:

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



The dovecot service starts the POP3 server

Please note that **dovecot** only reports that it started the IMAP server, but also starts the POP3 server.

Unlike SMTP, both IMAP and POP3 require connecting clients to authenticate using a user name and password. By default, passwords for both protocols are passed over the network unencrypted.

To configure SSL on **dovecot**:

- Edit the **/etc/pki/dovecot/dovecot-openssl.cnf** configuration file as you prefer. However, in a typical installation, this file does not require modification.
- Rename, move or delete the files **/etc/pki/dovecot/certs/dovecot.pem** and **/etc/pki/dovecot/private/dovecot.pem**.
- Execute the **/usr/libexec/dovecot/mkcert.sh** script which creates the **dovecot** self signed certificates. These certificates are copied in the **/etc/pki/dovecot/certs** and **/etc/pki/dovecot/private** directories. To implement the changes, restart **dovecot** by issuing the following command as root:

```
~]# systemctl restart dovecot
```

More details on **dovecot** can be found online at <http://www.dovecot.org>.

11.2. Email Program Classifications

In general, all email applications fall into at least one of three classifications. Each classification plays a specific role in the process of moving and managing email messages. While most users are only aware of the specific email program they use to receive and send messages, each one is important for ensuring that email arrives at the correct destination.

11.2.1. Mail Transport Agent

A *Mail Transport Agent (MTA)* transports email messages between hosts using SMTP. A message may involve several MTAs as it moves to its intended destination.

While the delivery of messages between machines may seem rather straightforward, the entire process of deciding if a particular MTA can or should accept a message for delivery is quite complicated. In addition, due to problems from spam, use of a particular MTA is usually restricted by the MTA's configuration or the access configuration for the network on which the MTA resides.

Many modern email client programs can act as an MTA when sending email. However, this action should not be confused with the role of a true MTA. The sole reason email client programs are capable of sending email like an MTA is because the host running the application does not have its own MTA. This is particularly true for email client programs on non-UNIX-based operating systems. However, these client programs only send outbound messages to an MTA they are authorized to use and do not directly deliver the message to the intended recipient's email server.

Since Fedora offers two MTAs, *Postfix* and *Sendmail*, email client programs are often not required to act as an MTA. Fedora also includes a special purpose MTA called *Fetchmail*.

For more information on Postfix, Sendmail, and Fetchmail, see [Section 11.3, “Mail Transport Agents”](#).

11.2.2. Mail Delivery Agent

A *Mail Delivery Agent (MDA)* is invoked by the MTA to file incoming email in the proper user's mailbox. In many cases, the MDA is actually a *Local Delivery Agent (LDA)*, such as **mail** or Procmail.

Any program that actually handles a message for delivery to the point where it can be read by an email client application can be considered an MDA. For this reason, some MTAs (such as Sendmail

and Postfix) can fill the role of an MDA when they append new email messages to a local user's mail spool file. In general, MDAs do not transport messages between systems nor do they provide a user interface; MDAs distribute and sort messages on the local machine for an email client application to access.

11.2.3. Mail User Agent

A *Mail User Agent (MUA)* is synonymous with an email client application. An MUA is a program that, at a minimum, allows a user to read and compose email messages. Many MUAs are capable of retrieving messages via the POP or IMAP protocols, setting up mailboxes to store messages, and sending outbound messages to an MTA.

MUAs may be graphical, such as **Evolution**, or have simple text-based interfaces, such as **Mutt**.

11.3. Mail Transport Agents

Fedora offers two primary MTAs: Postfix and Sendmail. Postfix is configured as the default MTA and Sendmail is considered deprecated. If required to switch the default MTA to Sendmail, you can either uninstall Postfix or use the following command as root to switch to Sendmail:

```
~]# alternatives --config mta
```

You can also use the following command to enable the desired service:

```
~]# systemctl enable service
```

Similarly, to disable the service, type the following at a shell prompt:

```
~]# systemctl disable service
```

For more information on how to manage system services in Fedora 26, see [Chapter 7, Services and Daemons](#).

11.3.1. Postfix

Originally developed at IBM by security expert and programmer Wietse Venema, Postfix is a Sendmail-compatible MTA that is designed to be secure, fast, and easy to configure.

To improve security, Postfix uses a modular design, where small processes with limited privileges are launched by a *master* daemon. The smaller, less privileged processes perform very specific tasks related to the various stages of mail delivery and run in a changed root environment to limit the effects of attacks.

Configuring Postfix to accept network connections from hosts other than the local computer takes only a few minor changes in its configuration file. Yet for those with more complex needs, Postfix provides a variety of configuration options, as well as third party add-ons that make it a very versatile and full-featured MTA.

The configuration files for Postfix are human readable and support upward of 250 directives. Unlike Sendmail, no macro processing is required for changes to take effect and the majority of the most commonly used options are described in the heavily commented files.

11.3.1.1. The Default Postfix Installation

The Postfix executable is **postfix**. This daemon launches all related processes needed to handle mail delivery.

Postfix stores its configuration files in the **/etc/postfix/** directory. The following is a list of the more commonly used files:

- **access** — Used for access control, this file specifies which hosts are allowed to connect to Postfix.
- **main.cf** — The global Postfix configuration file. The majority of configuration options are specified in this file.
- **master.cf** — Specifies how Postfix interacts with various processes to accomplish mail delivery.
- **transport** — Maps email addresses to relay hosts.

The **aliases** file can be found in the **/etc/** directory. This file is shared between Postfix and Sendmail. It is a configurable list required by the mail protocol that describes user ID aliases.



Configuring Postfix as a server for other clients

The default **/etc/postfix/main.cf** file does not allow Postfix to accept network connections from a host other than the local computer. For instructions on configuring Postfix as a server for other clients, see [Section 11.3.1.2, “Basic Postfix Configuration”](#).

Restart the **postfix** service after changing any options in the configuration files under the **/etc/postfix** directory in order for those changes to take effect. To do so, run the following command as root:

```
~]# systemctl restart postfix
```

11.3.1.2. Basic Postfix Configuration

By default, Postfix does not accept network connections from any host other than the local host. Perform the following steps as root to enable mail delivery for other hosts on the network:

- Edit the **/etc/postfix/main.cf** file with a text editor, such as **vi**.
- Uncomment the **mydomain** line by removing the hash sign (**#**), and replace *domain.tld* with the domain the mail server is servicing, such as **example.com**.
- Uncomment the **myorigin = \$mydomain** line.
- Uncomment the **myhostname** line, and replace *host.domain.tld* with the host name for the machine.
- Uncomment the **mydestination = \$myhostname, localhost.\$mydomain** line.
- Uncomment the **mynetworks** line, and replace *168.100.189.0/28* with a valid network setting for hosts that can connect to the server.
- Uncomment the **inet_interfaces = all** line.

- Comment the **inet_interfaces = localhost** line.
- Restart the **postfix** service.

Once these steps are complete, the host accepts outside emails for delivery.

Postfix has a large assortment of configuration options. One of the best ways to learn how to configure Postfix is to read the comments within the **/etc/postfix/main.cf** configuration file. Additional resources including information about Postfix configuration, SpamAssassin integration, or detailed descriptions of the **/etc/postfix/main.cf** parameters are available online at <http://www.postfix.org/>.

11.3.1.3. Using Postfix with LDAP

Postfix can use an LDAP directory as a source for various lookup tables (e.g.: **aliases**, **virtual**, **canonical**, etc.). This allows LDAP to store hierarchical user information and Postfix to only be given the result of LDAP queries when needed. By not storing this information locally, administrators can easily maintain it.

11.3.1.3.1. The **/etc/aliases** lookup example

The following is a basic example for using LDAP to look up the **/etc/aliases** file. Make sure your **/etc/postfix/main.cf** file contains the following:

```
alias_maps = hash:/etc/aliases, ldap:/etc/postfix/ldap-aliases.cf
```

Create a **/etc/postfix/ldap-aliases.cf** file if you do not have one already and make sure it contains the following:

```
server_host = ldap.example.com
search_base = dc=example, dc=com
```

where *ldap.example.com*, *example*, and *com* are parameters that need to be replaced with specification of an existing available LDAP server.



The **/etc/postfix/ldap-aliases.cf** file

The **/etc/postfix/ldap-aliases.cf** file can specify various parameters, including parameters that enable LDAP SSL and STARTTLS. For more information, see the **ldap_table(5)** man page.

For more information on LDAP, see [Section 12.1, “OpenLDAP”](#).

11.3.2. Sendmail

Sendmail's core purpose, like other MTAs, is to safely transfer email among hosts, usually using the SMTP protocol. Note that Sendmail is considered deprecated and users are encouraged to use Postfix when possible. See [Section 11.3.1, “Postfix”](#) for more information.

11.3.2.1. Purpose and Limitations

It is important to be aware of what Sendmail is and what it can do, as opposed to what it is not. In these days of monolithic applications that fulfill multiple roles, Sendmail may seem like the only application needed to run an email server within an organization. Technically, this is true, as Sendmail can spool mail to each users' directory and deliver outbound mail for users. However, most users actually require much more than simple email delivery. Users usually want to interact with their email using an MUA, that uses POP or IMAP, to download their messages to their local machine. Or, they may prefer a Web interface to gain access to their mailbox. These other applications can work in conjunction with Sendmail, but they actually exist for different reasons and can operate separately from one another.

It is beyond the scope of this section to go into all that Sendmail should or could be configured to do. With literally hundreds of different options and rule sets, entire volumes have been dedicated to helping explain everything that can be done and how to fix things that go wrong. See the [Section 11.6, “Additional Resources”](#) for a list of Sendmail resources.

This section reviews the files installed with Sendmail by default and reviews basic configuration changes, including how to stop unwanted email (spam) and how to extend Sendmail with the *Lightweight Directory Access Protocol (LDAP)*.

11.3.2.2. The Default Sendmail Installation

In order to use Sendmail, first ensure the *sendmail* package is installed on your system by running, as root:

```
~]# dnf install sendmail
```

In order to configure Sendmail, ensure the *sendmail-cf* package is installed on your system by running, as root:

```
~]# dnf install sendmail-cf
```

For more information on installing packages with DNF, see [Section 6.2.4, “Installing Packages”](#).

Before using Sendmail, the default MTA has to be switched from Postfix. For more information how to switch the default MTA refer to [Section 11.3, “Mail Transport Agents”](#).

The Sendmail executable is **sendmail**.

Sendmail's lengthy and detailed configuration file is **/etc/mail/sendmail.cf**. Avoid editing the **sendmail.cf** file directly. To make configuration changes to Sendmail, edit the **/etc/mail/sendmail.mc** file, back up the original **/etc/mail/sendmail.cf** file, and use the following alternatives to generate a new configuration file:

- Use the included makefile in **/etc/mail/** to create a new **/etc/mail/sendmail.cf** configuration file:

```
~]# make all -C /etc/mail/
```

All other generated files in **/etc/mail** (db files) will be regenerated if needed. The old makemap commands are still usable. The make command is automatically used whenever you start or restart the sendmail service.

More information on configuring Sendmail can be found in [Section 11.3.2.3, “Common Sendmail Configuration Changes”](#).

Various Sendmail configuration files are installed in the `/etc/mail/` directory including:

- **access** — Specifies which systems can use Sendmail for outbound email.
- **domaintable** — Specifies domain name mapping.
- **local-host-names** — Specifies aliases for the host.
- **mailertable** — Specifies instructions that override routing for particular domains.
- **virtusertable** — Specifies a domain-specific form of aliasing, allowing multiple virtual domains to be hosted on one machine.

Several of the configuration files in the `/etc/mail/` directory, such as **access**, **domaintable**, **mailertable** and **virtusertable**, must actually store their information in database files before Sendmail can use any configuration changes. To include any changes made to these configurations in their database files, run the following commands, as root:

```
~]# cd /etc/mail/  
~]# make all
```

This will update **virtusertable.db**, **access.db**, **domaintable.db**, **mailertable.db**, **sendmail.cf**, and **submit.cf**.

To update all the database files listed above and to update a custom database file, use a command in the following format:

```
make name.db all
```

where *name* represents the name of the custom database file to be updated.

To update a single database, use a command in the following format:

```
make name.db
```

where *name.db* represents the name of the database file to be updated.

You may also restart the `sendmail` service for the changes to take effect by running:

```
~]# systemctl restart sendmail
```

For example, to have all emails addressed to the **example.com** domain delivered to **bob@other-example.com**, add the following line to the **virtusertable** file:

```
@example.com bob@other-example.com
```

To finalize the change, the **virtusertable.db** file must be updated:

```
~]# make virtusertable.db all
```


Using the **all** option will result in the **virtusertable.db** and **access.db** being updated at the same time.

11.3.2.3. Common Sendmail Configuration Changes

When altering the Sendmail configuration file, it is best not to edit an existing file, but to generate an entirely new **/etc/mail/sendmail.cf** file.



Backup the sendmail.cf file before changing its content

Before replacing or making any changes to the **sendmail.cf** file, create a backup copy.

To add the desired functionality to Sendmail, edit the **/etc/mail/sendmail.mc** file as root. Once you are finished, restart the **sendmail** service and, if the **m4** package is installed, the **m4** macro processor will automatically generate a new **sendmail.cf** configuration file:

```
~]# systemctl restart sendmail
```



Configuring Sendmail as a server for other clients

The default **sendmail.cf** file does not allow Sendmail to accept network connections from any host other than the local computer. To configure Sendmail as a server for other clients, edit the **/etc/mail/sendmail.mc** file, and either change the address specified in the **Addr=** option of the **DAEMON_OPTIONS** directive from **127.0.0.1** to the IP address of an active network device or comment out the **DAEMON_OPTIONS** directive all together by placing **dnl** at the beginning of the line. When finished, regenerate **/etc/mail/sendmail.cf** by restarting the service:

```
~]# systemctl restart sendmail
```

The default configuration in Fedora works for most SMTP-only sites. However, it does not work for **UUCP** (*UNIX-to-UNIX Copy Protocol*) sites. If using UUCP mail transfers, the **/etc/mail/sendmail.mc** file must be reconfigured and a new **/etc/mail/sendmail.cf** file must be generated.

Consult the **/usr/share/sendmail-cf/README** file before editing any files in the directories under the **/usr/share/sendmail-cf/** directory, as they can affect the future configuration of the **/etc/mail/sendmail.cf** file.

11.3.2.4. Masquerading

One common Sendmail configuration is to have a single machine act as a mail gateway for all machines on the network. For example, a company may want to have a machine called **mail.example.com** that handles all of their email and assigns a consistent return address to all outgoing mail.

In this situation, the Sendmail server must masquerade the machine names on the company network so that their return address is **user@example.com** instead of **user@host.example.com**.

To do this, add the following lines to **/etc/mail/sendmail.mc**:

```
FEATURE(always_add_domain)dnl
FEATURE(`masquerade_entire_domain')dnl
FEATURE(`masquerade_envelope')dnl
FEATURE(`allmasquerade')dnl
MASQUERADE_AS(`example.com.')dnl
MASQUERADE_DOMAIN(`example.com.')dnl
MASQUERADE_AS(example.com)dnl
```

After generating a new **sendmail.cf** file using the **m4** macro processor, this configuration makes all mail from inside the network appear as if it were sent from **example.com**.

11.3.2.5. Stopping Spam

Email spam can be defined as unnecessary and unwanted email received by a user who never requested the communication. It is a disruptive, costly, and widespread abuse of Internet communication standards.

Sendmail makes it relatively easy to block new spamming techniques being employed to send junk email. It even blocks many of the more usual spamming methods by default. Main anti-spam features available in sendmail are *header checks*, *relaying denial* (default from version 8.9), *access database* and *sender information checks*.

For example, forwarding of SMTP messages, also called relaying, has been disabled by default since Sendmail version 8.9. Before this change occurred, Sendmail directed the mail host (**x.edu**) to accept messages from one party (**y.com**) and sent them to a different party (**z.net**). Now, however, Sendmail must be configured to permit any domain to relay mail through the server. To configure relay domains, edit the **/etc/mail/relay-domains** file and restart Sendmail

```
~]# systemctl restart sendmail
```

However users can also be sent spam from from servers on the Internet. In these instances, Sendmail's access control features available through the **/etc/mail/access** file can be used to prevent connections from unwanted hosts. The following example illustrates how this file can be used to both block and specifically allow access to the Sendmail server:

```
badspammer.com ERROR:550 "Go away and do not spam us anymore" tux.badspammer.com OK 10.0
RELAY
```

This example shows that any email sent from **badspammer.com** is blocked with a 550 RFC-821 compliant error code, with a message sent back. Email sent from the **tux.badspammer.com** sub-domain, is accepted. The last line shows that any email sent from the 10.0.*.* network can be relayed through the mail server.

Because the **/etc/mail/access.db** file is a database, use the **makemap** command to update any changes. Do this using the following command as root:

```
~]# makemap hash /etc/mail/access < /etc/mail/access
```

Message header analysis allows you to reject mail based on header contents. SMTP servers store information about an email's journey in the message header. As the message travels from one MTA

to another, each puts in a **Received** header above all the other **Received** headers. It is important to note that this information may be altered by spammers.

The above examples only represent a small part of what Sendmail can do in terms of allowing or blocking access. See the `/usr/share/sendmail-cf/README` file for more information and examples.

Since Sendmail calls the Procmail MDA when delivering mail, it is also possible to use a spam filtering program, such as SpamAssassin, to identify and file spam for users. See [Section 11.4.2.6, “Spam Filters”](#) for more information about using SpamAssassin.

11.3.2.6. Using Sendmail with LDAP

Using LDAP is a very quick and powerful way to find specific information about a particular user from a much larger group. For example, an LDAP server can be used to look up a particular email address from a common corporate directory by the user's last name. In this kind of implementation, LDAP is largely separate from Sendmail, with LDAP storing the hierarchical user information and Sendmail only being given the result of LDAP queries in pre-addressed email messages.

However, Sendmail supports a much greater integration with LDAP, where it uses LDAP to replace separately maintained files, such as `/etc/aliases` and `/etc/mail/virtusertables`, on different mail servers that work together to support a medium- to enterprise-level organization. In short, LDAP abstracts the mail routing level from Sendmail and its separate configuration files to a powerful LDAP cluster that can be leveraged by many different applications.

The current version of Sendmail contains support for LDAP. To extend the Sendmail server using LDAP, first get an LDAP server, such as **OpenLDAP**, running and properly configured. Then edit the `/etc/mail/sendmail.mc` to include the following:

```
LDAPROUTE_DOMAIN('yourdomain.com')dn1
FEATURE('ldap_routing')dn1
```

Advanced configuration

This is only for a very basic configuration of Sendmail with LDAP. The configuration can differ greatly from this depending on the implementation of LDAP, especially when configuring several Sendmail machines to use a common LDAP server.

Consult `/usr/share/sendmail-cf/README` for detailed LDAP routing configuration instructions and examples.

Next, recreate the `/etc/mail/sendmail.cf` file by running the `m4` macro processor and again restarting Sendmail. See [Section 11.3.2.3, “Common Sendmail Configuration Changes”](#) for instructions.

For more information on LDAP, see [Section 12.1, “OpenLDAP”](#).

11.3.3. Fetchmail

Fetchmail is an MTA which retrieves email from remote servers and delivers it to the local MTA. Many users appreciate the ability to separate the process of downloading their messages located on a

remote server from the process of reading and organizing their email in an MUA. Designed with the needs of dial-up users in mind, Fetchmail connects and quickly downloads all of the email messages to the mail spool file using any number of protocols, including POP3 and IMAP. It can even forward email messages to an SMTP server, if necessary.



Installing the fetchmail package

In order to use **Fetchmail**, first ensure the *fetchmail* package is installed on your system by running, as root:

```
-]# dnf install fetchmail
```

For more information on installing packages with DNF, see [Section 6.2.4, “Installing Packages”](#).

Fetchmail is configured for each user through the use of a **.fetchmailrc** file in the user's home directory. If it does not already exist, create the **.fetchmailrc** file in your home directory

Using preferences in the **.fetchmailrc** file, Fetchmail checks for email on a remote server and downloads it. It then delivers it to port 25 on the local machine, using the local MTA to place the email in the correct user's spool file. If Procmail is available, it is launched to filter the email and place it in a mailbox so that it can be read by an MUA.

11.3.3.1. Fetchmail Configuration Options

Although it is possible to pass all necessary options on the command line to check for email on a remote server when executing Fetchmail, using a **.fetchmailrc** file is much easier. Place any desired configuration options in the **.fetchmailrc** file for those options to be used each time the **fetchmail** command is issued. It is possible to override these at the time Fetchmail is run by specifying that option on the command line.

A user's **.fetchmailrc** file contains three classes of configuration options:

- *global options* — Gives Fetchmail instructions that control the operation of the program or provide settings for every connection that checks for email.
- *server options* — Specifies necessary information about the server being polled, such as the host name, as well as preferences for specific email servers, such as the port to check or number of seconds to wait before timing out. These options affect every user using that server.
- *user options* — Contains information, such as user name and password, necessary to authenticate and check for email using a specified email server.

Global options appear at the top of the **.fetchmailrc** file, followed by one or more server options, each of which designate a different email server that Fetchmail should check. User options follow server options for each user account checking that email server. Like server options, multiple user options may be specified for use with a particular server as well as to check multiple email accounts on the same server.

Server options are called into service in the **.fetchmailrc** file by the use of a special option verb, **poll** or **skip**, that precedes any of the server information. The **poll** action tells Fetchmail to use this server option when it is run, which checks for email using the specified user options.

Any server options after a **skip** action, however, are not checked unless this server's host name is specified when Fetchmail is invoked. The **skip** option is useful when testing configurations in the **.fetchmailrc** file because it only checks skipped servers when specifically invoked, and does not affect any currently working configurations.

The following is an example of a **.fetchmailrc** file:

```
set postmaster "user1"
set bouncemail

poll pop.domain.com proto pop3
    user 'user1' there with password 'secret' is user1 here

poll mail.domain2.com
    user 'user5' there with password 'secret2' is user1 here
    user 'user7' there with password 'secret3' is user1 here
```

In this example, the global options specify that the user is sent email as a last resort (**postmaster** option) and all email errors are sent to the postmaster instead of the sender (**bouncemail** option). The **set** action tells Fetchmail that this line contains a global option. Then, two email servers are specified, one set to check using POP3, the other for trying various protocols to find one that works. Two users are checked using the second server option, but all email found for any user is sent to **user1**'s mail spool. This allows multiple mailboxes to be checked on multiple servers, while appearing in a single MUA inbox. Each user's specific information begins with the **user** action.

Omitting the password from the configuration

Users are not required to place their password in the **.fetchmailrc** file. Omitting the **with password 'password'** section causes Fetchmail to ask for a password when it is launched.

Fetchmail has numerous global, server, and local options. Many of these options are rarely used or only apply to very specific situations. The **fetchmail** man page explains each option in detail, but the most common ones are listed in the following three sections.

11.3.3.2. Global Options

Each global option should be placed on a single line after a **set** action.

- **daemon seconds** — Specifies daemon-mode, where Fetchmail stays in the background. Replace *seconds* with the number of seconds Fetchmail is to wait before polling the server.
- **postmaster** — Specifies a local user to send mail to in case of delivery problems.
- **syslog** — Specifies the log file for errors and status messages. By default, this is **/var/log/maillog**.

11.3.3.3. Server Options

Server options must be placed on their own line in **.fetchmailrc** after a **poll** or **skip** action.

- **auth auth-type** — Replace *auth-type* with the type of authentication to be used. By default, **password** authentication is used, but some protocols support other types of authentication,

including **kerberos_v5**, **kerberos_v4**, and **ssh**. If the **any** authentication type is used, Fetchmail first tries methods that do not require a password, then methods that mask the password, and finally attempts to send the password unencrypted to authenticate to the server.

- **interval *number*** — Polls the specified server every *number* of times that it checks for email on all configured servers. This option is generally used for email servers where the user rarely receives messages.
- **port *port-number*** — Replace *port-number* with the port number. This value overrides the default port number for the specified protocol.
- **proto *protocol*** — Replace *protocol* with the protocol, such as **pop3** or **imap**, to use when checking for messages on the server.
- **timeout *seconds*** — Replace *seconds* with the number of seconds of server inactivity after which Fetchmail gives up on a connection attempt. If this value is not set, a default of **300** seconds is used.

11.3.3.4. User Options

User options may be placed on their own lines beneath a server option or on the same line as the server option. In either case, the defined options must follow the **user** option (defined below).

- **fetchall** — Orders Fetchmail to download all messages in the queue, including messages that have already been viewed. By default, Fetchmail only pulls down new messages.
- **fetchlimit *number*** — Replace *number* with the number of messages to be retrieved before stopping.
- **flush** — Deletes all previously viewed messages in the queue before retrieving new messages.
- **limit *max-number-bytes*** — Replace *max-number-bytes* with the maximum size in bytes that messages are allowed to be when retrieved by Fetchmail. This option is useful with slow network links, when a large message takes too long to download.
- **password '*password*'** — Replace *password* with the user's password.
- **preconnect "*command*"** — Replace *command* with a command to be executed before retrieving messages for the user.
- **postconnect "*command*"** — Replace *command* with a command to be executed after retrieving messages for the user.
- **ssl** — Activates SSL encryption.
- **user "*username*"** — Replace *username* with the username used by Fetchmail to retrieve messages. *This option must precede all other user options.*

11.3.3.5. Fetchmail Command Options

Most Fetchmail options used on the command line when executing the **fetchmail** command mirror the **.fetchmailrc** configuration options. In this way, Fetchmail may be used with or without a configuration file. These options are not used on the command line by most users because it is easier to leave them in the **.fetchmailrc** file.

There may be times when it is desirable to run the **fetchmail** command with other options for a particular purpose. It is possible to issue command options to temporarily override a **.fetchmailrc** setting that is causing an error, as any options specified at the command line override configuration file options.

11.3.3.6. 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.

11.3.3.7. 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.

11.3.4. Mail Transport Agent (MTA) Configuration

A *Mail Transport Agent* (MTA) is essential for sending email. A *Mail User Agent* (MUA) such as **Evolution**, **Thunderbird**, and **Mutt**, is used to read and compose email. When a user sends an email from an MUA, the message is handed off to the MTA, which sends the message through a series of MTAs until it reaches its destination.

Even if a user does not plan to send email from the system, some automated tasks or system programs might use the **mail** command to send email containing log messages to the root user of the local system.

Fedora 26 provides two MTAs: Postfix and Sendmail. If both are installed, Postfix is the default MTA. Note that Sendmail is considered deprecated in MAJOROS;.

11.4. Mail Delivery Agents

Fedora 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.

11.4.1. 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**.

- **INCLUDEDRC** — 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=$HOME/Msgs
INCLUDEDRC=$MAILDIR/lists.rc
INCLUDEDRC=$MAILDIR/spam.rc
```

To turn off Procmail filtering of email lists but leaving spam control in place, comment out the first **INCLUDEDRC** 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 **INCLUDEDRC** 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.

11.4.2. 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 11.4.2.5, “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 11.4.2.4, "Special Conditions and Actions"](#) for more information.

11.4.2.1. 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.

11.4.2.2. 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.

11.4.2.3. 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.

11.4.2.4. 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.

11.4.2.5. 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 **MAILEDIR** 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.



Sending messages to `/dev/null`

Be certain that rules are working as intended before sending messages to `/dev/null` for permanent deletion. If a recipe inadvertently catches unintended messages, and those messages disappear, it becomes difficult to troubleshoot the rule.

A better solution is to point the recipe's action to a special mailbox, which can be checked from time to time to look for false positives. Once satisfied that no messages are accidentally being matched, delete the mailbox and direct the action to send the messages to `/dev/null`.

The following recipe grabs email sent from a particular mailing list and places it in a specified folder.

```
:0:
* ^(From|Cc|To).*tux-lug
tuxlug
```

Any messages sent from the **tux-lug@domain.com** mailing list are placed in the **tuxlug** mailbox automatically for the MUA. Note that the condition in this example matches the message if it has the mailing list's email address on the **From**, **Cc**, or **To** lines.

Consult the many Procmail online resources available in [Section 11.6, “Additional Resources”](#) for more detailed and powerful recipes.

11.4.2.6. Spam Filters

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

This is particularly true when Procmail is used in conjunction with SpamAssassin. When used together, these two applications can quickly identify spam emails, and sort or destroy them.

SpamAssassin uses header analysis, text analysis, blacklists, a spam-tracking database, and self-learning Bayesian spam analysis to quickly and accurately identify and tag spam.



Installing the spamassassin package

In order to use **SpamAssassin**, first ensure the *spamassassin* package is installed on your system by running, as root:

```
~]# dnf install spamassassin
```

For more information on installing packages with DNF, see [Section 6.2.4, “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:


```
INCLUDEDRC=/etc/mail/spamassassin/spamassassin-default.rc
```

The `/etc/mail/spamassassin/spamassassin-default.rc` contains a simple Procmail rule that activates SpamAssassin for all incoming email. If an email is determined to be spam, it is tagged in the header as such and the title is prepended with the following pattern:

```
*****SPAM*****
```

The message body of the email is also prepended with a running tally of what elements caused it to be diagnosed as spam.

To file email tagged as spam, a rule similar to the following can be used:

```
:0 Hw * ^X-Spam-Status: Yes spam
```

This rule files all email tagged in the header as spam into a mailbox called **spam**.

Since SpamAssassin is a Perl script, it may be necessary on busy servers to use the binary SpamAssassin daemon (`spamd`) and the client application (**spamc**). Configuring SpamAssassin this way, however, requires `root` access to the host.

To start the `spamd` daemon, type the following command:

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

To start the SpamAssassin daemon when the system is booted, run:

```
systemctl enable spamassassin.service
```

See [Chapter 7, Services and Daemons](#) for more information on how to configure services in Fedora.

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`:

```
INCLUDEDRC=/etc/mail/spamassassin/spamassassin-spamc.rc
```

11.5. Mail User Agents

Fedora 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.

11.5.1. Securing Communication

Popular MUAs included with Fedora, 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.

11.5.1.1. 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.

11.5.1.2. 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.



Avoid using self-signed certificates

Self-signed certificates should be used for testing purposes only. Any server used in a production environment should use an SSL certificate signed 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
```

Issue the following command to restart the **dovecot** daemon:

```
~]# systemctl restart dovecot
```

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 Fedora 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.

Installing the stunnel package

In order to use **stunnel**, first ensure the *stunnel* package is installed on your system by running, as root:

```
~]# dnf install stunnel
```

For more information on installing packages with DNF, see [Section 6.2.4, “Installing Packages”](#).

To create a self-signed SSL certificate, change to the **/etc/pki/tls/certs/** directory, and type the following command:

```
certs]# make stunnel.pem
```

Answer all of the questions to complete the process.

Once the certificate is generated, create an **stunnel** configuration file, for example **/etc/stunnel/mail.conf**, with the following content:

```
cert = /etc/pki/tls/certs/stunnel.pem

[pop3s]
accept  = 995
connect = 110

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

Once you start **stunnel** with the created configuration file using the **stunnel /etc/stunnel/mail.conf** command, it will be possible to use an IMAP or a POP email client and connect to the email server using SSL encryption.

For more information on **stunnel**, see the **stunnel(8)** man page or the documents in the **/usr/share/doc/stunnel/** directory.

11.6. Additional Resources

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

11.6.1. 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/` — Contains a large amount of information on how to configure Postfix.
- `/usr/share/doc/fetchmail/` — Contains a full list of Fetchmail features in the **FEATURES** file and an introductory **FAQ** document.
- `/usr/share/doc/procmail/` — 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.

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/` — Contains a large amount of information pertaining to SpamAssassin.

11.6.2. Useful Websites

- <http://www.sendmail.org/> — Offers a thorough technical breakdown of Sendmail features, documentation and configuration examples.
- <http://www.sendmail.com/> — Contains news, interviews and articles concerning Sendmail, including an expanded view of the many options available.
- <http://www.postfix.org/> — The Postfix project home page contains a wealth of information about Postfix. The mailing list is a particularly good place to look for information.
- <http://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.uwasa.fi/~ts/info/proctips.html> — Contains dozens of tips that make using Procmail much easier. Includes instructions on how to test **.procmailrc** files and use Procmail scoring to decide if a particular action should be taken.
- <http://www.spamassassin.org/> — The official site of the SpamAssassin project.

11.6.3. Related Books

- *Sendmail Milters: 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 Procmal, 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.

Directory Servers

12.1. 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.

12.1.1. 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.



Using Mozilla NSS

The OpenLDAP suite in Fedora 26 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](http://www.openldap.org/faq/index.cgi?file=1514)¹.

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.

12.1.1.1. LDAP Terminology

The following is a list of LDAP-specific terms that are used within this chapter:

¹ <http://www.openldap.org/faq/index.cgi?file=1514>

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.

12.1.1.2. OpenLDAP Features

OpenLDAP suite provides a number of important features:

- *LDAPv3 Support* — Many of the changes in the protocol since LDAP version 2 are designed to make LDAP more secure. Among other improvements, this includes the support for Simple Authentication and Security Layer (SASL), Transport Layer Security (TLS), and Secure Sockets Layer (SSL) protocols.
- *LDAP Over IPC* — The use of inter-process communication (IPC) enhances security by eliminating the need to communicate over a network.
- *IPv6 Support* — OpenLDAP is compliant with Internet Protocol version 6 (IPv6), the next generation of the Internet Protocol.
- *LDIFv1 Support* — OpenLDAP is fully compliant with LDIF version 1.
- *Updated C API* — The current C API improves the way programmers can connect to and use LDAP directory servers.
- *Enhanced Standalone LDAP Server* — This includes an updated access control system, thread pooling, better tools, and much more.

12.1.1.3. OpenLDAP Server Setup

The typical steps to set up an LDAP server on Fedora are as follows:

1. Install the OpenLDAP suite. See [Section 12.1.2, “Installing the OpenLDAP Suite”](#) for more information on required packages.
2. Customize the configuration as described in [Section 12.1.3, “Configuring an OpenLDAP Server”](#).
3. Start the `slapd` service as described in [Section 12.1.5, “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.

12.1.2. Installing the OpenLDAP Suite

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

Table 12.1. List of OpenLDAP packages

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

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

Table 12.2. List of commonly installed additional LDAP packages

Package	Description
<i>nss-pam-ldapd</i>	A package containing <code>nsldap</code> , a local LDAP name service that allows a user to perform local LDAP queries.
<i>mod_ldap</i>	A package containing the <code>mod_authnz_ldap</code> and <code>mod_ldap</code> modules. The <code>mod_authnz_ldap</code> module is the LDAP authorization module for the Apache HTTP Server. This module can authenticate users' credentials against an LDAP directory, and can enforce access control based on the user name, full DN, group membership, an arbitrary attribute, or a complete filter string. The <code>mod_ldap</code> module contained in the same package provides a configurable shared memory cache, to avoid repeated directory access across many HTTP requests, and also support for SSL/TLS.

To install these packages, use the `dnf` command in the following form:

```
dnf install package...
```


For example, to perform the basic LDAP server installation, type the following at a shell prompt as root:

```
~]# dnf 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 Fedora, see [Section 6.2.4, “Installing Packages”](#).

12.1.2.1. Overview of OpenLDAP Server Utilities

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

Table 12.3. List of OpenLDAP server utilities

Command	Description
slapacl	Allows you to check the access to a list of attributes.
slapadd	Allows you to add entries from an LDIF file to an LDAP directory.
slapauth	Allows you to check a list of IDs for authentication and authorization permissions.
slapcat	Allows you to pull entries from an LDAP directory in the default format and save them in an LDIF file.
slapdn	Allows you to check a list of Distinguished Names (DNs) based on available schema syntax.
slapindex	Allows you to re-index the <i>slapd</i> directory based on the current content. Run this utility whenever you change indexing options in the configuration file.
slappasswd	Allows you to create an encrypted user password to be used with the ldapmodify utility, or in the <i>slapd</i> configuration file.
slapschema	Allows you to check the compliance of a database with the corresponding schema.
slaptest	Allows you to check the LDAP server configuration.

For a detailed description of these utilities and their usage, see the corresponding manual pages as referred to in [the section called “Installed Documentation”](#).



Make sure the files have correct owner

Although only root can run **slapadd**, the *slapd* service runs as the *ldap* user. Because of this, the directory server is unable to modify any files created by **slapadd**. To correct this issue, after running the **slapadd** utility, type the following at a shell prompt:

```
chown -R ldap:ldap /var/lib/ldap
```




Stop slapd before using these utilities

To preserve the data integrity, stop the slapd service before using **slapadd**, **slapcat**, or **slapindex**. You can do so by typing the following at a shell prompt as root:

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

For more information on how to start, stop, restart, and check the current status of the slapd service, see [Section 12.1.5, “Running an OpenLDAP Server”](#).

12.1.2.2. 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 12.4. List of OpenLDAP client utilities

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

With the exception of **ldapsearch**, each of these utilities is more easily used by referencing a file containing the changes to be made rather than typing a command for each entry to be changed within an LDAP directory. The format of such a file is outlined in the man page for each utility.

12.1.2.3. 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 Fedora. Popular applications that can access directories in a read-only mode include **Mozilla Thunderbird**, **Evolution**, or **Ekiga**.

12.1.3. Configuring an OpenLDAP Server

By default, the OpenLDAP configuration is stored in the **/etc/openldap/** directory. The following table highlights the most important directories and files within this directory:

Table 12.5. 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 ldapadd , ldapsearch , Evolution , etc.
<code>/etc/openldap/slapd.d/</code>	The directory containing the <code>slapd</code> configuration.

Note that OpenLDAP no longer reads its configuration from the `/etc/openldap/slapd.conf` file. Instead, it uses a configuration database located in the `/etc/openldap/slapd.d/` directory. If you have an existing **slapd.conf** file from a previous installation, you can convert it to the new format by running the following command as root:

```
~]# 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 12.1.2.1, “Overview of OpenLDAP Server Utilities”](#).



Do not edit LDIF files directly

An error in an LDIF file can render the `slapd` service unable to start. Because of this, it is strongly advised that you avoid editing the LDIF files within the `/etc/openldap/slapd.d/` directly.

12.1.3.1. 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 12.6, “Available olcAllows options”](#). The default option is **bind_v2**.

Table 12.6. Available olcAllows options

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

Option	Description
proxy_authz_anon	Enables processing of anonymous proxy authorization control.

Example 12.1. 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 12.2. 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 12.3. 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 12.7, “Available `olcDisallows` options”](#). No features are disabled by default.

Table 12.7. Available `olcDisallows` options

Option	Description
bind_anon	Disables the acceptance of anonymous bind requests.
bind_simple	Disables the simple bind authentication mechanism.
tls_2_anon	Disables the enforcing of an anonymous session when the STARTTLS command is received.
tls_authc	Disallows the STARTTLS command when authenticated.

Example 12.4. 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 12.5. 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 12.6. 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 12.7. 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 12.8. Using the `olcWriteTimeout` directive

```
olcWriteTimeout: 180
```

12.1.3.2. Changing the Database-Specific Configuration

By default, the OpenLDAP server uses Berkeley DB (BDB) as a database back end. The configuration for this database is stored in the `/etc/openldap/slapd.d/cn=config/olcDatabase={1}bdb.ldif` file. The following directives are commonly used in a database-specific configuration:

`olcReadOnly`

The **`olcReadOnly`** directive allows you to use the database in a read-only mode. It takes the following form:

```
olcReadOnly: boolean
```

It accepts either **TRUE** (enable the read-only mode), or **FALSE** (enable modifications of the database). The default option is **FALSE**.

Example 12.9. Using the `olcReadOnly` directive

```
olcReadOnly: TRUE
```

`olcRootDN`

The **`olcRootDN`** directive allows you to specify the user that is unrestricted by access controls or administrative limit parameters set for operations on the LDAP directory. It takes the following form:

```
olcRootDN: distinguished_name
```

It accepts a *Distinguished Name* (DN). The default option is **cn=Manager, dn=my-domain, dc=com**.

Example 12.10. Using the `olcRootDN` directive

```
olcRootDN: cn=root, dn=example, dn=com
```

`olcRootPW`

The **`olcRootPW`** directive allows you to set a password for the user that is specified using the **`olcRootDN`** directive. It takes the following form:

```
olcRootPW: password
```

It accepts either a plain text string, or a hash. To generate a hash, type the following at a shell prompt:

```
~]$ slappaswd
```



```
New password:
Re-enter new password:
{SSHA}WczWsyPEnMchFf1GRTweq2q7XJcvmSxD
```

Example 12.11. 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 12.12. Using the `olcSuffix` directive

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

12.1.3.3. Extending Schema

Since OpenLDAP 2.3, the `/etc/openldap/slapd.d/` directory also contains LDAP definitions that were previously located in `/etc/openldap/schema/`. It is possible to extend the schema used by OpenLDAP to support additional attribute types and object classes using the default schema files as a guide. However, this task is beyond the scope of this chapter. For more information on this topic, see <http://www.openldap.org/doc/admin/schema.html>.

12.1.3.4. Establishing a Secure Connection

OpenLDAP clients and servers can be secured using the Transport Layer Security (TLS) framework. TLS is a cryptographic protocol designed to provide communication security over the network. As noted above, OpenLDAP suite in Fedora uses Mozilla NSS as the TLS implementation.

To establish a secure connection using TLS, obtain the required certificates as described in [How do I use TLS/SSL with Mozilla NSS](#)². Then, a number of options must be configured on both the client and the server. At a minimum, a server must be configured with the Certificate Authority (CA) certificates and also its own server certificate and private key. The clients must be configured with the name of the file containing all the trusted CA certificates.

Typically, a server only needs to sign a single CA certificate. A client may want to connect to a variety of secure servers, therefore it is common to specify a list of several trusted CAs in its configuration.

Server Configuration

This section lists global configuration directives for `slapd` that need to be specified in the `/etc/openldap/slapd.d/cn=config.ldif` file on an OpenLDAP server in order to establish TLS.

² <http://www.openldap.org/faq/index.cgi?file=1514>

While the old style configuration uses a single file, normally installed as `/usr/local/etc/openldap/slapd.conf`, the new style uses a slapd backend database to store the configuration. The configuration database normally resides in the `/usr/local/etc/openldap/slapd.d/` directory.

The following directives are also valid for establishing SSL. In addition to TLS directives, you need to enable a port dedicated to SSL on the server side – typically it is port 636. To do so, edit the `/etc/sysconfig/slapd` file and append the `ldaps:///` string to the list of URLs specified with the `SLAPD_URLS` directive.

olcTLSCACertificateFile

The **olcTLSCACertificateFile** directive specifies the file encoded with Privacy-Enhanced Mail (PEM) schema that contains trusted CA certificates. The directive takes the following form:

```
olcTLSCACertificateFile: path
```

Replace *path* either with a path to the CA certificate file, or, if you use Mozilla NSS, with a certificate name.

olcTLSCACertificatePath

The **olcTLSCACertificatePath** directive specifies the path to a directory containing individual CA certificates in separate files. This directory must be specially managed with the OpenSSL **c_rehash** utility that generates symbolic links with the hashed names that point to the actual certificate files. In general, it is simpler to use the **olcTLSCACertificateFile** directive instead.

If Mozilla NSS is used, **olcTLSCACertificatePath** accepts a path to the Mozilla NSS database (as shown in [Example 12.13](#), “Using **olcTLSCACertificatePath** with Mozilla NSS”). In such a case, **c_rehash** is not needed.

The directive takes the following form:

```
olcTLSCACertificatePath: path
```

Replace *path* with a path to the directory containing the CA certificate files, or with a path to a Mozilla NSS database file.

Example 12.13. Using **olcTLSCACertificatePath with Mozilla NSS**

With Mozilla NSS, the **olcTLSCACertificatePath** directive specifies the path of the directory containing the NSS certificate and key database files. For example:

```
olcTLSCACertificatePath: sql:/home/nssdb/sharednssdb
```

The **certutil** command is used to add a CA certificate to these NSS database files:

```
certutil -d sql:/home/nssdb/sharednssdb -A -n "CA_certificate" -t CT,, -a -i certificate.pem
```

The above command adds a CA certificate stored in a PEM-formatted file named *certificate.pem*. The **-d** option specifies the database directory containing the certificate and key database files, the **-n** option sets a name for the certificate, **-t CT,,** means that the certificate is trusted to be used in TLS clients and servers. The **-A** option adds an existing certificate to a certificate database, the **-a** option allows the use of ASCII format for input or output, and the **-i** option passes the **certificate.pem** input file to the command.

olcTLSCertificateFile

The **olcTLSCertificateFile** directive specifies the file that contains the slapd server certificate. The directive takes the following form:

```
olcTLSCertificateFile: path
```

Replace *path* with a path to the slapd server certificate file, or, if you use Mozilla NSS, with a certificate name.

[Example 12.14. Using olcTLSCertificateFile with Mozilla NSS](#)

When using Mozilla NSS with certificate and key database files specified with the **olcTLSCACertificatePath** directive, **olcTLSCertificateFile** is used to specify the name of the certificate to use. First, execute the following command to view a list of certificates available in your NSS database file:

```
certutil -d sql:/home/nssdb/sharednssdb -L
```

Select a certificate from the list and pass its name to **olcTLSCertificateFile**. For example:

```
olcTLSCertificateFile slapd_cert
```

olcTLSCertificateKeyFile

The **olcTLSCertificateKeyFile** directive specifies the file that contains the private key that matches the certificate stored in the file specified with **olcTLSCertificateFile**. Note that the current implementation does not support encrypted private keys, and therefore the containing file must be sufficiently protected. The directive takes the following form:

```
olcTLSCertificateKeyFile: path
```

Replace *path* with a path to the private key file if you use PEM certificates. When using Mozilla NSS, *path* stands for the name of a file that contains the password for the key for the certificate specified with the **olcTLSCertificateFile** directive (see [Example 12.15, “Using olcTLSCertificateKeyFile with Mozilla NSS”](#)).

[Example 12.15. Using olcTLSCertificateKeyFile with Mozilla NSS](#)

When using Mozilla NSS, this directive specifies the name of a file that contains the password for the key for the certificate specified with **olcTLSCertificateFile**:

```
olcTLSCertificateKeyFile: slapd_cert_key
```

The **modutil** command can be used to turn off password protection or to change the password for NSS database files. For example:

```
modutil -dbdir sql:/home/nssdb/sharednssdb -changePW
```

Client Configuration

Specify the following directives in the **/etc/openldap/ldap.conf** configuration file on the client system. Most of these directives are parallel to the server configuration options. Directives in **/etc/openldap/ldap.conf** are configured on a system-wide basis, however, individual users may override them in their **~/.ldaprc** files.

The same directives can be used to establish an SSL connection. The **ldaps://** string must be used instead of **ldap://** in OpenLDAP commands such as **ldapsearch**. This forces commands to use the default port for SSL, port 636, configured on the server.

TLS_CACERT

The **TLS_CACERT** directive specifies a file containing certificates for all of the Certificate Authorities the client will recognize. This is equivalent to the **olcTLSCACertificateFile** directive on a server. **TLS_CACERT** should always be specified before **TLS_CACERTDIR** in **/etc/openldap/ldap.conf**. The directive takes the following form:

```
TLS_CACERT path
```

Replace *path* with a path to the CA certificate file.

TLS_CACERTDIR

The **TLS_CACERTDIR** directive specifies the path to a directory that contains Certificate Authority certificates in separate files. As with **olcTLSCACertificatePath** on a server, the specified directory must be managed with the OpenSSL **c_rehash** utility. Path to Mozilla NSS database file is also accepted, **c_rehash** is not needed in such case. The directive takes the following form:

```
TLS_CACERTDIR directory
```

Replace *directory* with a path to the directory containing CA certificate files. With Mozilla NSS, *directory* stands for a path to the certificate or key database file.

TLS_CERT

The **TLS_CERT** specifies the file that contains a client certificate. This directive can only be specified in a user's **~/ .ldaprc** file. With Mozilla NSS, this directive specifies the name of the certificate to be chosen from the database specified with the aforementioned **TLS_CACERTDIR** directive. The directive takes the following form:

```
TLS_CERT path
```

Replace *path* with a path to the client certificate file, or with a name of a certificate from the NSS database.

TLS_KEY

The **TLS_KEY** specifies the file that contains the private key that matches the certificate stored in the file specified with the **TLS_CERT** directive. As with **olcTLSCertificateFile** on a server, encrypted key files are not supported, so the file itself must be carefully protected. This option is only configurable in a user's **~/ .ldaprc** file.

When using Mozilla NSS, **TLS_KEY** specifies the name of a file that contains the password for the private key that protects the certificate specified with the **TLS_CERT** directive. Similarly to the **olcTLSCertificateKeyFile** directive on a server (see [Example 12.15, "Using olcTLSCertificateKeyFile with Mozilla NSS"](#)), you can use the **modutil** command to manage this password.

The **TLS_KEY** directive takes the following form:

```
TLS_KEY path
```

Replace *path* with a path to the client certificate file or with a name of the password file in the NSS database.

12.1.3.5. Setting Up Replication

Replication is the process of copying updates from one LDAP server (*provider*) to one or more other servers or clients (*consumers*). A provider replicates directory updates to consumers, the received updates can be further propagated by the consumer to other servers, so a consumer can also act simultaneously as a provider. Also, a consumer does not have to be an LDAP server, it may be just an LDAP client. In OpenLDAP, you can use several replication modes, most notable are *mirror* and *sync*. For more information on OpenLDAP replication modes, see the *OpenLDAP Software Administrator's Guide* installed with *openldap-servers* package (see [the section called "Installed Documentation"](#)).

To enable a chosen replication mode, use one of the following directives in `/etc/openldap/slapd.d/` on both provider and consumers.

olcMirrorMode

The **olcMirrorMode** directive enables the mirror replication mode. It takes the following form:

```
olcMirrorMode on
```

This option needs to be specified both on provider and consumers. Also a **serverID** must be specified along with **syncrepl** options. Find a detailed example in the 18.3.4. *MirrorMode* section of the *OpenLDAP Software Administrator's Guide* (see [the section called "Installed Documentation"](#)).

olcSyncrepl

The **olcSyncrepl** directive enables the sync replication mode. It takes the following form:

```
olcSyncrepl on
```

The sync replication mode requires a specific configuration on both the provider and the consumers. This configuration is thoroughly described in the 18.3.1. *Syncrepl* section of the *OpenLDAP Software Administrator's Guide* (see [the section called "Installed Documentation"](#)).

12.1.3.6. Loading Modules and Backends

You can enhance the `slapd` service with dynamically loaded modules. Support for these modules must be enabled with the `--enable-modules` option when configuring `slapd`. Modules are stored in files with the `.la` extension:

```
module_name.la
```

Backends store or retrieve data in response to LDAP requests. Backends may be compiled statically into `slapd`, or when module support is enabled, they may be dynamically loaded. In the latter case, the following naming convention is applied:

```
back_backend_name.la
```

To load a module or a backend, use the following directive in `/etc/openldap/slapd.d/`:

olcModuleLoad

The **olcModuleLoad** directive specifies a dynamically loadable module to load. It takes the following form:


```
olcModuleLoad: module
```

Here, *module* stands either for a file containing the module, or a backend, that will be loaded.

12.1.4. SELinux Policy for Applications Using LDAP

SELinux is an implementation of a mandatory access control mechanism in the Linux kernel. By default, SELinux prevents applications from accessing an OpenLDAP server. To enable authentication through LDAP, which is required by several applications, the `allow_yppbind` SELinux Boolean needs to be enabled. Certain applications also demand an enabled `authlogin_nsswitch_use_ldap` Boolean in this scenario. Execute the following commands to enable the aforementioned Booleans:

```
~]# setsebool -P allow_yppbind=1
```

```
~]# setsebool -P authlogin_nsswitch_use_ldap=1
```

The `-P` option makes this setting persistent across system reboots. See the [Red Hat Enterprise Linux 7 SELinux User's and Administrator's Guide](#)³ for more detailed information about SELinux.

12.1.5. 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 7, Services and Daemons](#).

12.1.5.1. Starting the Service

To start the `slapd` service in the current session, type the following at a shell prompt as root:

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

To configure the service to start automatically at the boot time, use the following command as root:

```
~]# systemctl enable slapd.service
```

See [Chapter 7, Services and Daemons](#) for more information on how to configure services in Fedora.

12.1.5.2. Stopping the Service

To stop the running `slapd` service in the current session, type the following at a shell prompt as root:

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

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

³ https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/SELinux_Users_and_Administrators_Guide/


```
~]# systemctl disable slapd.service  
rm '/etc/systemd/system/multi-user.target.wants/slapd.service'
```

See [Chapter 7, Services and Daemons](#) for more information on how to configure services in Fedora.

12.1.5.3. Restarting the Service

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

```
~]# systemctl restart slapd.service
```

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

12.1.5.4. Verifying the Service Status

To verify that the slapd service is running, type the following at a shell prompt:

```
~]$ systemctl is-active slapd.service  
active
```

12.1.6. 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 12.1.2, “Installing the OpenLDAP Suite”](#) and [Section 12.1.3, “Configuring an OpenLDAP Server”](#). On a client, type the following at a shell prompt as root:

```
~]# dnf install openldap openldap-clients nss-pam-ldapd
```

12.1.6.1. 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 as root:

```
~]# dnf 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 12.8, “Commonly used LDAP migration scripts”](#).

Table 12.8. Commonly used LDAP migration scripts

Existing Name Service	Is LDAP Running?	Script to Use
/etc flat files	yes	migrate_all_online.sh
/etc flat files	no	migrate_all_offline.sh
NetInfo	yes	migrate_all_netinfo_online.sh
NetInfo	no	migrate_all_netinfo_offline.sh
NIS (YP)	yes	migrate_all_nis_online.sh
NIS (YP)	no	migrate_all_nis_offline.sh

For more information on how to use these scripts, see the **README** and the **migration-tools.txt** files in the `/usr/share/doc/migrationtools/` directory.

12.1.7. 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*.

Installed Documentation

The following documentation is installed with the *openldap-servers* package:

- `/usr/share/doc/openldap-servers/guide.html` — A copy of the *OpenLDAP Software Administrator's Guide*.
- `/usr/share/doc/openldap-servers/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

- `ldapadd(1)` — The manual page for the **ldapadd** command describes how to add entries to an LDAP directory.
- `ldapdelete(1)` — The manual page for the **ldapdelete** command describes how to delete entries within an LDAP directory.
- `ldapmodify(1)` — The manual page for the **ldapmodify** command describes how to modify entries within an LDAP directory.
- `ldapsearch(1)` — The manual page for the **ldapsearch** command describes how to search for entries within an LDAP directory.
- `ldappasswd(1)` — The manual page for the **ldappasswd** command describes how to set or change the password of an LDAP user.
- `ldapcompare(1)` — Describes how to use the **ldapcompare** tool.

- `ldapwhoami(1)` — Describes how to use the **ldapwhoami** tool.
- `ldapmodrdn(1)` — Describes how to modify the RDNs of entries.

Server Applications

- `slapd(8C)` — Describes command line options for the LDAP server.

Administrative Applications

- `slapadd(8C)` — Describes command line options used to add entries to a **slapd** database.
- `slapcat(8C)` — Describes command line options used to generate an LDIF file from a **slapd** database.
- `slapindex(8C)` — Describes command line options used to regenerate an index based upon the contents of a **slapd** database.
- `slappasswd(8C)` — Describes command line options used to generate user passwords for LDAP directories.

Configuration Files

- `ldap.conf(5)` — The manual page for the **ldap.conf** file describes the format and options available within the configuration file for LDAP clients.
- `slapd-config(5)` — Describes the format and options available within the `/etc/openldap/slapd.d` configuration directory.

Online Documentation

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

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

<http://www.kingsmountain.com/ldapRoadmap.shtml>

Jeff Hodges' *LDAP Roadmap & FAQ* containing links to several useful resources and emerging news concerning the LDAP protocol.

<http://www.ldapman.org/articles/>

A collection of articles that offer a good introduction to LDAP, including methods to design a directory tree and customizing directory structures.

<http://www.padl.com/>

A website of developers of several useful LDAP tools.

12.1.8. 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.

File and Print Servers

This chapter guides you through the installation and configuration of **Samba**, an open source implementation of the *Server Message Block* (SMB) and *common Internet file system* (CIFS) protocol, and **vsftpd**, the primary FTP server shipped with Fedora. Additionally, it explains how to use the **Printer** tool to configure printers.

13.1. 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.

Installing the samba package

In order to use **Samba**, first ensure the *samba* package is installed on your system by running, as root:

```
~]# dnf install samba
```

For more information on installing packages with DNF, see [Section 6.2.4, “Installing Packages”](#).

13.1.1. 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/Windows Server 2012

What Samba cannot do:

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

13.1.2. 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.

Obtaining a list of utilities that are shipped with Samba

See [Section 13.1.12, “Samba Distribution Programs”](#) for a list of utilities included in the Samba distribution.

13.1.3. Connecting to a Samba Share

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

Procedure 13.1. 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**.

An icon appears for each available SMB workgroup or domain on the network.

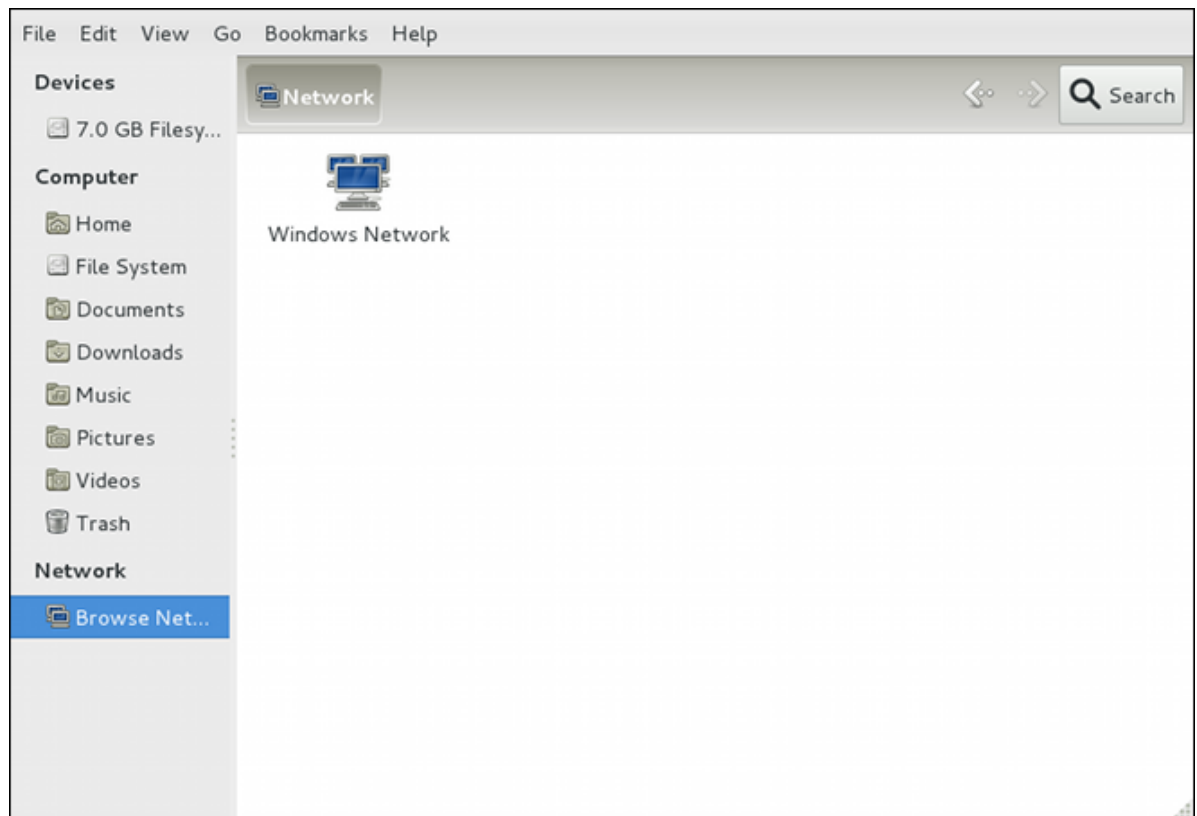


Figure 13.1. SMB Workgroups in Nautilus

2. Double-click one of the workgroup or domain icon to view a list of computers within the workgroup or domain.
3. 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 13.2. Connecting to a Samba Share Using the Command Line

1. 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.

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

13.1.4. 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.



Installing cifs-utils package

The **mount.cifs** utility is a separate RPM (independent from Samba). In order to use **mount.cifs**, first ensure the *cifs-utils* package is installed on your system by running, as root:

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

For more information on installing packages with DNF, see [Section 6.2.4, “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.



CIFS servers that require plain text passwords

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.

13.1.5. 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.

13.1.5.1. 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/>.

13.1.5.2. Command-Line Configuration

Samba uses `/etc/samba/smb.conf` as its configuration file. If you change this configuration file, the changes do not take effect until you restart the Samba daemon with the following command, as root:

```
~]# systemctl restart smb.service
```

To specify the Windows workgroup and a brief description of the Samba server, edit the following lines in your `/etc/samba/smb.conf` file:


```
workgroup = WORKGROUPNAME
server string = BRIEF COMMENT ABOUT SERVER
```

Replace *WORKGROUPNAME* with the name of the Windows workgroup to which this machine should belong. The *BRIEF COMMENT ABOUT SERVER* is optional and is used as the Windows comment about the Samba system.

To create a Samba share directory on your Linux system, add the following section to your **/etc/samba/smb.conf** file (after modifying it to reflect your needs and your system):

Example 13.1. 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.

13.1.5.3. 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
```

13.1.6. Starting and Stopping Samba

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

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



Setting up a domain member server

To set up a domain member server, you must first join the domain or Active Directory using the **net join** command *before* starting the smb service. Also, it is recommended to run `winbind` before `smbd`.

To stop the server, type the following command in a shell prompt, as root:

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


The **restart** option is a quick way of stopping and then starting Samba. This is the most reliable way to make configuration changes take effect after editing the configuration file for Samba. Note that the restart option starts the daemon even if it was not running originally.

To restart the server, type the following command in a shell prompt, as root:

```
~]# systemctl restart smb.service
```

The **condrestart** (*conditional restart*) option only starts smb on the condition that it is currently running. This option is useful for scripts, because it does not start the daemon if it is not running.



Applying the changes to the configuration

When the **/etc/samba/smb.conf** file is changed, Samba automatically reloads it after a few minutes. Issuing a manual **restart** or **reload** is just as effective.

To conditionally restart the server, type the following command, as root:

```
~]# systemctl try-restart smb.service
```

A manual reload of the **/etc/samba/smb.conf** file can be useful in case of a failed automatic reload by the smb service. To ensure that the Samba server configuration file is reloaded without restarting the service, type the following command, as root:

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

By default, the smb service does *not* start automatically at boot time. To configure Samba to start at boot time, type the following at a shell prompt as root:

```
~]# systemctl enable smb.service
```

See [Chapter 7, Services and Daemons](#) for more information regarding this tool.

13.1.7. 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.

13.1.7.1. 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 13.1.8, “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 13.2. 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.

Do not use anonymous read/write servers

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 13.3. 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 13.4. 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 13.5. 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
```

13.1.7.2. 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 13.3. 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 [Fedora 26 Managing Single Sign-On and Smart Cards](#)¹.

3. 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.



The security option

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.

¹ https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Managing_Smart_Cards/Using_Kerberos.html

Example 13.6. 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.



Make sure you join the domain before starting Samba

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.

13.1.7.3. 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.



A mixed Samba/Windows domain controller environment

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 tdbsam

The simplest and most common implementation of a Samba PDC uses the new default tdbsam password database back end. Replacing the aging smbpasswd back end, tdbsam has numerous improvements that are explained in more detail in [Section 13.1.9, “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 tdbsam password database back end.

Example 13.7. 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 `t db sam` follow these steps:

1. Adjust the **smb.conf** configuration file as shown in [Example 13.7, “An Example Configuration of Primary Domain Controller \(PDC\) Using `t db sam`”](#).
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.



Limitations of the tdbsam authentication back end

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.

13.1.8. 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 has been removed from Samba. Configurations containing this mode need to be migrated to use user-level security. User-level security can be implemented in one of three different ways. The different ways of implementing a security level are called *security modes*.

13.1.8.1. 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. To configure a Samba guest share without using the *security = share* parameter, follow the procedure below:

Procedure 13.4. 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
...
```


13.1.8.2. 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 has been removed from Samba. Configurations containing `security = share` should be updated to use user-level security. Follow the steps in [Procedure 13.4, “Configuring Samba Guest Shares”](#) to avoid using the `security = share` directive.

13.1.9. 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](https://access.redhat.com/documentation/en-US/Red_Hat_Directory_Server/9.0/html/Deployment_Guide/index.html)². For more information on LDAP, see [Section 12.1, “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 12.1.3.3, “Extending Schema”](#) for directions on how to do this.

Make sure the `openldap-servers` package is installed

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 root:

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

13.1.10. 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.

13.1.10.1. 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 13.1.5, “Configuring a Samba Server”](#)).

² https://access.redhat.com/documentation/en-US/Red_Hat_Directory_Server/9.0/html/Deployment_Guide/index.html

13.1.10.2. 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 13.8. An Example Configuration of WINS Server

```
[global]
wins support = yes
```

Using WINS

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.

13.1.11. 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 Fedora, CUPS (Common UNIX Print System) is the recommended printing system due to its close integration with Samba.

13.1.11.1. Simple `smb.conf` Settings

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

Example 13.9. 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.

13.1.12. Samba Distribution Programs

net

```
net <protocol> <function> <misc_options> <target_options>
```

The *net* utility is similar to the *net* utility used for Windows and MS-DOS. The first argument is used to specify the protocol to use when executing a command. The **protocol** option can be **ads**, **rap**, or **rpc** for specifying the type of server connection. Active Directory uses **ads**, Win9x/NT3 uses **rap**, and Windows NT4/2000/2003/2008 uses **rpc**. If the protocol is omitted, *net* automatically tries to determine it.

The following example displays a list of the available shares for a host named wakko:

```
~]$ net -l share -S wakko
Password:
Enumerating shared resources (exports) on remote server:
Share name  Type      Description
-----
data        Disk      Wakko data share
tmp         Disk      Wakko tmp share
IPC$        IPC       IPC Service (Samba Server)
ADMIN$      IPC       IPC Service (Samba Server)
```

The following example displays a list of Samba users for a host named wakko:

```
~]$ net -l user -S wakko
root password:
User name    Comment
-----
andriusb     Documentation
joe          Marketing
```


lisa	Sales
------	-------

nmblookup

```
nmblookup <options> <netbios_name>
```

The nmblookup program resolves NetBIOS names into IP addresses. The program broadcasts its query on the local subnet until the target machine replies.

The following example displays the IP address of the NetBIOS name **trek**:

```
~]$ nmblookup trek
querying trek on 10.1.59.255
10.1.56.45 trek<00>
```

pdbedit

```
pdbedit <options>
```

The pdbedit program manages accounts located in the SAM database. All back ends are supported including smbpasswd, LDAP, and the tdb database library.

The following are examples of adding, deleting, and listing users:

```
~]$ pdbedit -a kristin
new password:
retype new password:
Unix username:      kristin
NT username:
Account Flags:      [U                ]
User SID:           S-1-5-21-1210235352-3804200048-1474496110-2012
Primary Group SID:  S-1-5-21-1210235352-3804200048-1474496110-2077
Full Name: Home Directory:      \\wakko\kristin
HomeDir Drive:
Logon Script:
Profile Path:       \\wakko\kristin\profile
Domain:            WAKKO
Account desc:
Workstations: Munged
dial:
Logon time:         0
Logoff time:        Mon, 18 Jan 2038 22:14:07 GMT
Kickoff time:       Mon, 18 Jan 2038 22:14:07 GMT
Password last set:  Thu, 29 Jan 2004 08:29:28
GMT Password can change: Thu, 29 Jan 2004 08:29:28 GMT
Password must change: Mon, 18 Jan 2038 22:14:07 GMT
~]$ pdbedit -v -L kristin
Unix username:      kristin
NT username:
Account Flags:      [U                ]
User SID:           S-1-5-21-1210235352-3804200048-1474496110-2012
Primary Group SID:  S-1-5-21-1210235352-3804200048-1474496110-2077
Full Name:
Home Directory:     \\wakko\kristin
```



```
HomeDir Drive:
Logon Script:
Profile Path:      \\wakko\kristin\profile
Domain:           WAKKO
Account desc:
Workstations: Munged
dial:
Logon time:        0
Logoff time:       Mon, 18 Jan 2038 22:14:07 GMT
Kickoff time:      Mon, 18 Jan 2038 22:14:07 GMT
Password last set: Thu, 29 Jan 2004 08:29:28 GMT
Password can change: Thu, 29 Jan 2004 08:29:28 GMT
Password must change: Mon, 18 Jan 2038 22:14:07 GMT
~]$ pdbedit -L
andriusb:505:
joe:503:
lisa:504:
kristin:506:
~]$ pdbedit -x joe
~]$ pdbedit -L
andriusb:505: lisa:504: kristin:506:
```

rpcclient

```
rpcclient <server> <options>
```

The `rpcclient` program issues administrative commands using Microsoft RPCs, which provide access to the Windows administration graphical user interfaces (GUIs) for systems management. This is most often used by advanced users that understand the full complexity of Microsoft RPCs.

smbcacs

```
smbcacs <server/share> <filename> <options>
```

The `smbcacs` 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.

13.1.13. 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)

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 Fedora specific, some concepts may apply.
- https://wiki.samba.org/index.php/User_Documentation — Samba 4.x official documentation.
- <http://samba.org/samba/archives.html>³ — 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.org⁴, that use the NNTP protocol are also available. This an alternative to receiving mailing list emails.

13.2. FTP

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 into the remote host or 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, as well as configuration options for the primary FTP server shipped with Fedora, **vsftpd**.

13.2.1. The File Transfer Protocol

However, because FTP is so prevalent on the Internet, it is often required to share files to the public. System administrators, therefore, should be aware of the FTP protocol's unique characteristics.

13.2.1.1. Multiple Ports, Multiple Modes

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

³ <http://us1.samba.org/samba/archives.html>

⁴ <http://www.gmane.org/>

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 to protect 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 resolves 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 13.2.5.8, “Network Options”](#) for more information about limiting passive ports.

13.2.2. FTP Servers

Fedora ships with two different FTP servers:

- **proftpd** - A fast, stable, and highly configurable FTP server.
- **vsftpd** — A fast, secure FTP daemon which is the preferred FTP server for Fedora. The remainder of this section focuses on **vsftpd**.

13.2.2.1. vsftpd

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 Fedora, 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 run with the minimal privileges required for the task.
- *Tasks requiring elevated privileges are handled by processes with the minimal privilege necessary* — By leveraging compatibilities found in the **libcapp** 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 directory **/var/ftp/** 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 below 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 are received over a socket, and the validity of any information from child processes is checked before being acted on.
- *Most interaction with FTP clients is 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 allows the attacker access to the shared files.

13.2.3. Files Installed with **vsftpd**

The **vsftpd** RPM installs the daemon (**/usr/sbin/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/rc.d/init.d/vsftpd** — The *initialization script (initscript)* used by the **systemctl** command to start, stop, or reload **vsftpd**. See [Section 13.2.4, “Starting and Stopping vsftpd”](#) for more information about using this script.
- **/etc/pam.d/vsftpd** — The Pluggable Authentication Modules (PAM) configuration file for **vsftpd**. This file specifies the requirements a user must meet to login to the FTP server. For more information on PAM, refer to the *Using Pluggable Authentication Modules (PAM)* chapter of the *Fedora 26 Managing Single Sign-On and Smart Cards* guide.
- **/etc/vsftpd/vsftpd.conf** — The configuration file for **vsftpd**. See [Section 13.2.5, “vsftpd Configuration Options”](#) for a list of important options contained within this file.
- **/etc/vsftpd/ftpusers** — A list of users not allowed to log into **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 usernames 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.

13.2.4. Starting and Stopping **vsftpd**

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

To start the server, as root type:


```
systemctl start vsftpd.service
```

To stop the server, as root type:

```
systemctl stop vsftpd.service
```

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 root type:

```
systemctl restart vsftpd.service
```

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.

To conditionally restart the server, as root type:

```
systemctl condrestart vsftpd.service
```

By default, the **vsftpd** service does *not* start automatically at boot time. To configure the **vsftpd** service to start at boot time, use a service manager such as **systemctl**. See [Chapter 7, Services and Daemons](#) for more information on how to configure services in Fedora.

13.2.4.1. Starting Multiple Copies of vsftpd

Sometimes one computer is used to serve multiple FTP domains. This is a technique called *multihoming*. One way to multihome using **vsftpd** is by running multiple copies of the daemon, each with its own configuration file.

To do this, first assign all relevant IP addresses to network devices or alias network devices on the system. For more information about configuring network devices, device aliases, and additional information about network configuration scripts, refer to the [Fedora Networking Guide](#)⁵.

Next, the DNS server for the FTP domains must be configured to reference the correct machine. For information about BIND and its configuration files, refer to the [Fedora Networking Guide](#)⁶.

If there is more configuration files present in the **/etc/vsftpd** directory, calling **systemctl start vsftpd.service** results in the **/etc/rc.d/init.d/vsftpd** initscript starting the same number of processes as the number of configuration files. Each configuration file must have a unique name in the **/etc/vsftpd/** directory and must be readable and writable only by root.

13.2.5. vsftpd Configuration Options

Although **vsftpd** may not offer the level of customization other widely available FTP servers have, it offers enough options to fill most administrator's 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:

⁵ <https://docs.fedoraproject.org/en-US/Fedora/networking>

⁶ <https://docs.fedoraproject.org/en-US/Fedora/networking>


```
directive=value
```

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



Do not use spaces

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 sign (#) and are ignored by the daemon.

For a complete list of all directives available, refer to the man page for **vsftpd.conf**.



Securing the vsftpd service

For an overview of ways to secure **vsftpd**, see the [Red Hat Enterprise Linux 7 Security Guide](#)⁷.

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 **vsftpd**'s configuration file are set to their default value.

13.2.5.1. Daemon Options

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

- **listen** — When enabled, **vsftpd** runs in stand-alone mode. Fedora sets this value to **YES**. This directive cannot be used in conjunction with the **listen_ipv6** directive.

The default value is **NO**.

- **listen_ipv6** — When enabled, **vsftpd** runs in stand-alone mode, but listens only to IPv6 sockets. 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, refer to the *Using Pluggable Authentication Modules (PAM)* chapter of the Red Hat Enterprise Linux 6 *Managing 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**.

⁷ https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/

13.2.5.2. Log In Options and Access Controls

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

- **anonymous_enable** — When enabled, anonymous users are allowed to log in. The usernames **anonymous** and **ftp** are accepted.

The default value is **YES**.

See [Section 13.2.5.3, “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 which 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 the **/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 into the system.

The default value is **YES**.

See [Section 13.2.5.4, “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**. Note, in Fedora, the value is set to **vsftpd**.

- The default value is **NO**. Note, in Fedora, the value is set to **YES**.

- **userlist_deny** — When used in conjunction with the **userlist_enable** directive and set to **NO**, all local users are denied access unless the username 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, the 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**, however under Fedora the value is set to **YES**.

- **userlist_file** — Specifies the file referenced by **vsftpd** when the **userlist_enable** directive is enabled.

The default value is **/etc/vsftpd/user_list** and is created during installation.

13.2.5.3. Anonymous User Options

The following lists directives which 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 which 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 which 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 are accepted. This is a convenient way to offer 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**.

13.2.5.4. Local User Options

The following lists directives which characterize the way local users access the server. To use these options, the **local_enable** directive must be set to **YES**.

- **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**.



Avoid enabling the **chroot_local_user** option

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 username 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 **022**.

- **passwd_chroot_enable** — When enabled in conjunction with the **chroot_local_user** directive, **vsftpd** change-roots local users based on the occurrence of the `/./` 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 name of local system users that contain specific setting for that user. Any directive in the user's configuration file overrides those found in `/etc/vsftpd/vsftpd.conf`.

There is no default value for this directive.

13.2.5.5. Directory Options

The following lists directives which 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**. Note, in Fedora, the value is set to **YES**.

- **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 usernames and group names are used in place of UID and GID entries. Enabling this option may slow 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**.

13.2.5.6. File Transfer Options

The following lists directives which 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 **YES**.

13.2.5.7. Logging Options

The following lists directives which 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**. Note, in Fedora, the value is set to **YES**.

- **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**. Note, in Fedora, the value is set to **YES**.



Maintaining compatibility with older log file formats

To maintain compatibility with log files written by the older **wu-ftpd** FTP server, the **xferlog_std_format** directive is set to **YES** under Fedora. 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 sign (#), or delete the line entirely.

13.2.5.8. Network Options

The following lists directives which affect 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**. Note, in Fedora, the value is set to **YES**.

- **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.



Running multiple copies of vsftpd

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 13.2.4.1, “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.



Running multiple copies of vsftpd

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 13.2.4.1, “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 data is transferred for local users logged into 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 of clients allowed to connected from the same source IP address.

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

- **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 connects are allowed.

The default value is **YES**.

- **pasv_max_port** — Specifies the highest possible port sent to the 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 the 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 **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.



Avoid enabling the pasv_promiscuous option

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**.

13.2.6. Additional Resources

For more information about **vsftpd**, refer to the following resources.

13.2.6.1. Installed Documentation

- The **/usr/share/doc/vsftpd/** directory — This directory contains a **README** with basic information about the software. The **TUNING** file contains basic performance tuning tips and the **SECURITY/** directory contains information about the security model employed by **vsftpd**.

- **vsftpd** related man pages — There are a number of man pages for the daemon and configuration files. The following lists some of the more important man pages.

Server Applications

- **man vsftpd** — Describes available command line options for **vsftpd**.

Configuration Files

- **man vsftpd.conf** — Contains a detailed list of options available within the configuration file for **vsftpd**.
- **man 5 hosts_access** — Describes the format and options available within the TCP wrappers configuration files: **hosts.allow** and **hosts.deny**.

13.2.6.2. Useful Websites

- <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.
- <http://www.ietf.org/rfc/rfc0959.txt> — The original *Request for Comments (RFC)* of the FTP protocol from the IETF.

13.3. Printer Configuration

The **Printers** 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 Fedora version that used CUPS, the upgrade process preserved the configured printers.

Using the CUPS web application or command-line tools

You can perform the same and additional operations on printers directly from the CUPS web application or command line. To access the application, in a web browser, go to <http://localhost:631/>. For CUPS manuals refer to the links on the **Home** tab of the web site.

13.3.1. Starting the Printers Configuration Tool

With the **Printers** configuration tool you can perform various operations on existing printers and set up new printers. You can also use CUPS directly (go to <http://localhost:631/> to access the CUPS web application).

To start the **Printers** configuration tool if using the GNOME desktop, press the **Super** key to enter the Activities Overview, type **Printers**, and then press **Enter**. The **Printers** configuration tool appears. The **Super** key appears in a variety of guises, depending on the keyboard and other hardware, but often as either the Windows or Command key, and typically to the left of the **Spacebar**.

The **Printers** window depicted in [Figure 13.2, “Printers Configuration window”](#) appears.

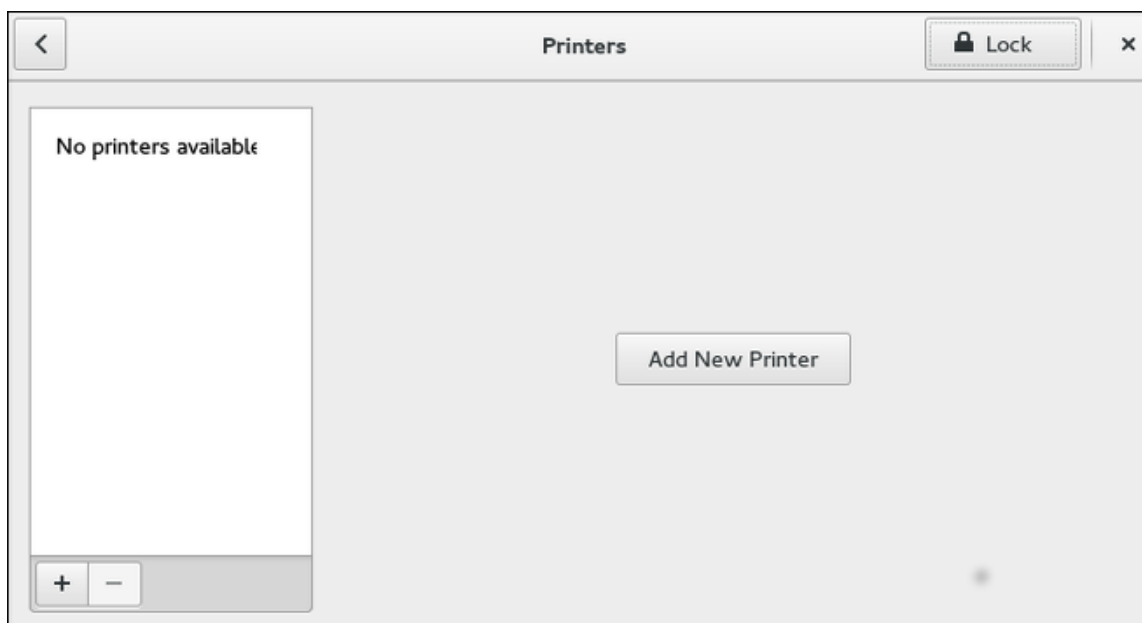


Figure 13.2. Printers Configuration window

13.3.2. 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 an administrator or the root user 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 Printers configuration tool (refer to [Section 13.3.1, “Starting the Printers Configuration Tool”](#)).
2. Select **Unlock** to enable changes to be made. In the **Authentication Required** box, type an administrator or the root user password and confirm.
3. Select the plus sign to open the **Add a New Printer** dialog. Select the printer from the list or enter its address below.

13.3.3. Adding a Local Printer

Follow this procedure to add a local printer connected with other than a serial port:

1. Open the Add a New Printer dialog (refer to [Section 13.3.2, “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 **Enter URI**

URI (for example file:/dev/lp0)

for **Serial Port**

Baud Rate

Parity

Data Bits

Flow Control

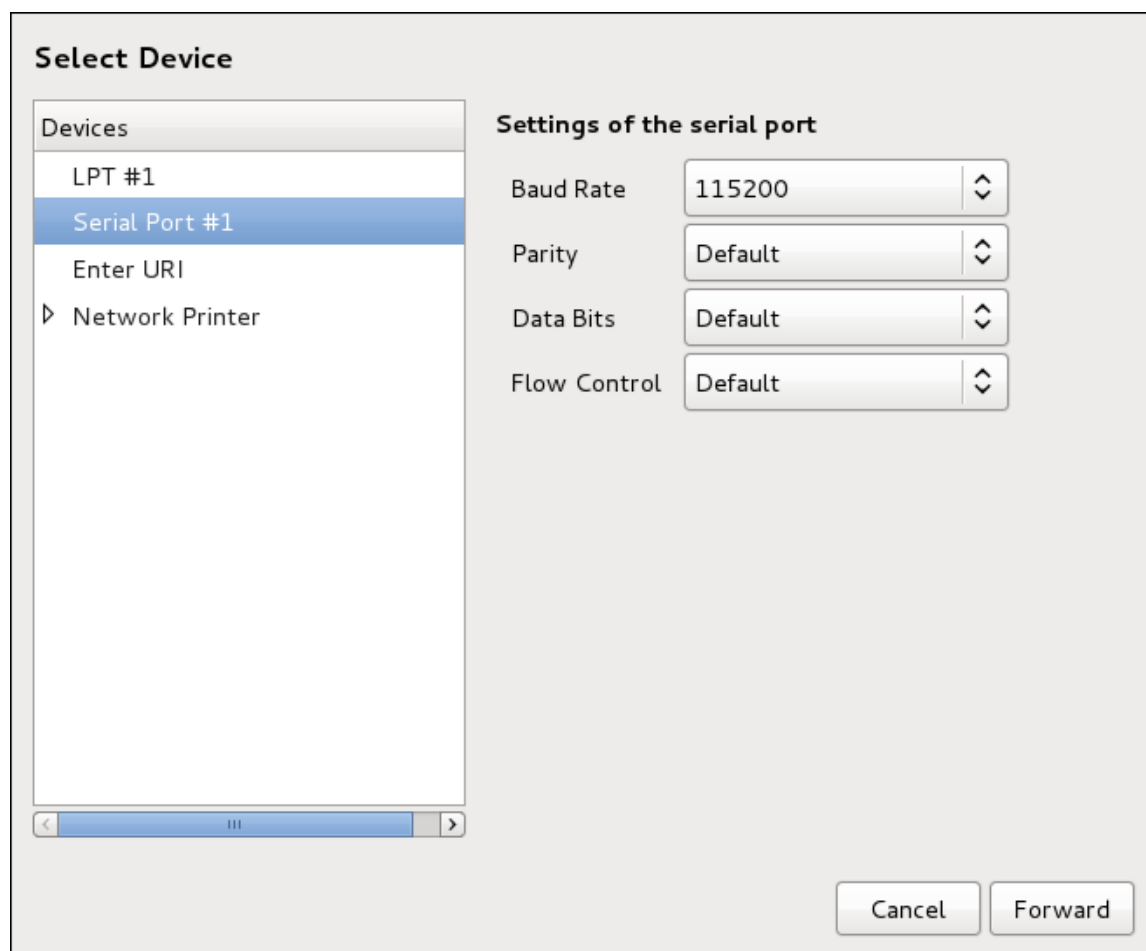


Figure 13.3. Adding a local printer

4. Click **Forward**.
5. Select the printer model. See [Section 13.3.8, “Selecting the Printer Model and Finishing”](#) for details.

13.3.4. Adding an AppSocket/HP JetDirect printer

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

1. Open the Add a New Printer dialog (refer to [Section 13.3.1, “Starting the Printers Configuration Tool”](#)).
2. In the list on the left, select **Network Printer** → **AppSocket/HP JetDirect**.

- 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).

Select Device

Devices

- LPT #1
- Serial Port #1
- Enter URI
- ▼ Network Printer
 - Find Network Printer
 - AppSocket/HP JetDirect**
 - Internet Printing Protocol (
 - Internet Printing Protocol (
 - Windows Printer via SAMBA

Location of the network printer

Host:

Port number:

Cancel Forward

Figure 13.4. Adding a JetDirect printer

- Click **Forward**.
- Select the printer model. See [Section 13.3.8, “Selecting the Printer Model and Finishing”](#) for details.

13.3.5. Adding an IPP Printer

An IPP printer is a printer attached to a different system on the same TCP/IP network. The system this printer is attached to may either be running CUPS or simply configured to use IPP.

If a firewall is enabled on the printer server, then the firewall must be configured to allow incoming TCP connections on port **631**. Note that the CUPS browsing protocol allows client machines to discover shared CUPS queues automatically. To enable this, the firewall on the client machine must be configured to allow incoming UDP packets on port **631**.

Follow this procedure to add an IPP printer:

1. Open the Printers dialog (refer to [Section 13.3.2, “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).

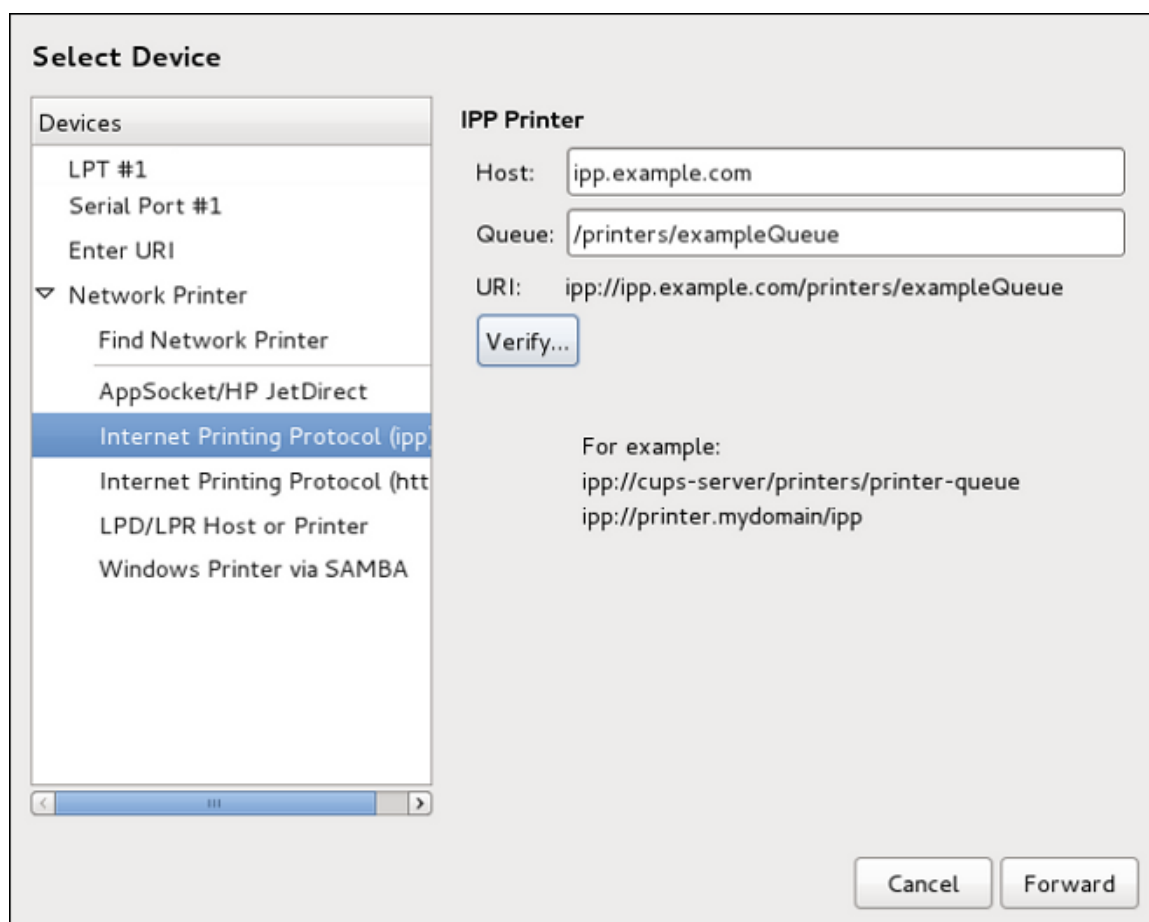


Figure 13.5. Adding an IPP printer

4. Optionally, click **Verify** to detect the printer.
5. Click **Forward** to continue.
6. Select the printer model. See [Section 13.3.8, “Selecting the Printer Model and Finishing”](#) for details.

13.3.6. Adding an LPD/LPR Host or Printer

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

1. Open the New Printer dialog (refer to [Section 13.3.2, “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).

The screenshot shows a window titled "Select Device". On the left, under the "Devices" header, there is a list: "LPT #1", "Serial Port #1", "Enter URI", "Network Printer" (expanded), "Find Network Printer", "AppSocket/HP JetDirect", "Internet Printing Protocol (ipp)", "Internet Printing Protocol (https)", "LPD/LPR Host or Printer" (highlighted), and "Windows Printer via SAMBA". On the right, under the heading "Location of the LPD network printer", there is a "Host:" label followed by a text box containing "lenore.example.com" and a dropdown arrow, and a "Probe" button with a magnifying glass icon. Below that is a "Queue:" label followed by a text box containing "/printers/" and a dropdown arrow. At the bottom right are "Cancel" and "Forward" buttons.

Figure 13.6. Adding an LPD/LPR printer

4. Click **Forward** to continue.
5. Select the printer model. See [Section 13.3.8, “Selecting the Printer Model and Finishing”](#) for details.

13.3.7. Adding a Samba (SMB) printer

Follow this procedure to add a Samba printer:

Installing the samba-client package

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:

```
dnf install samba-client
```

For more information on installing packages with DNF, refer to [Section 6.2.4, “Installing Packages”](#).

1. Open the New Printer dialog (refer to [Section 13.3.2, “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 13.7, “Adding a SMB printer”](#), the *computer name* is **dellbox** and the *printer share* is **r2**.

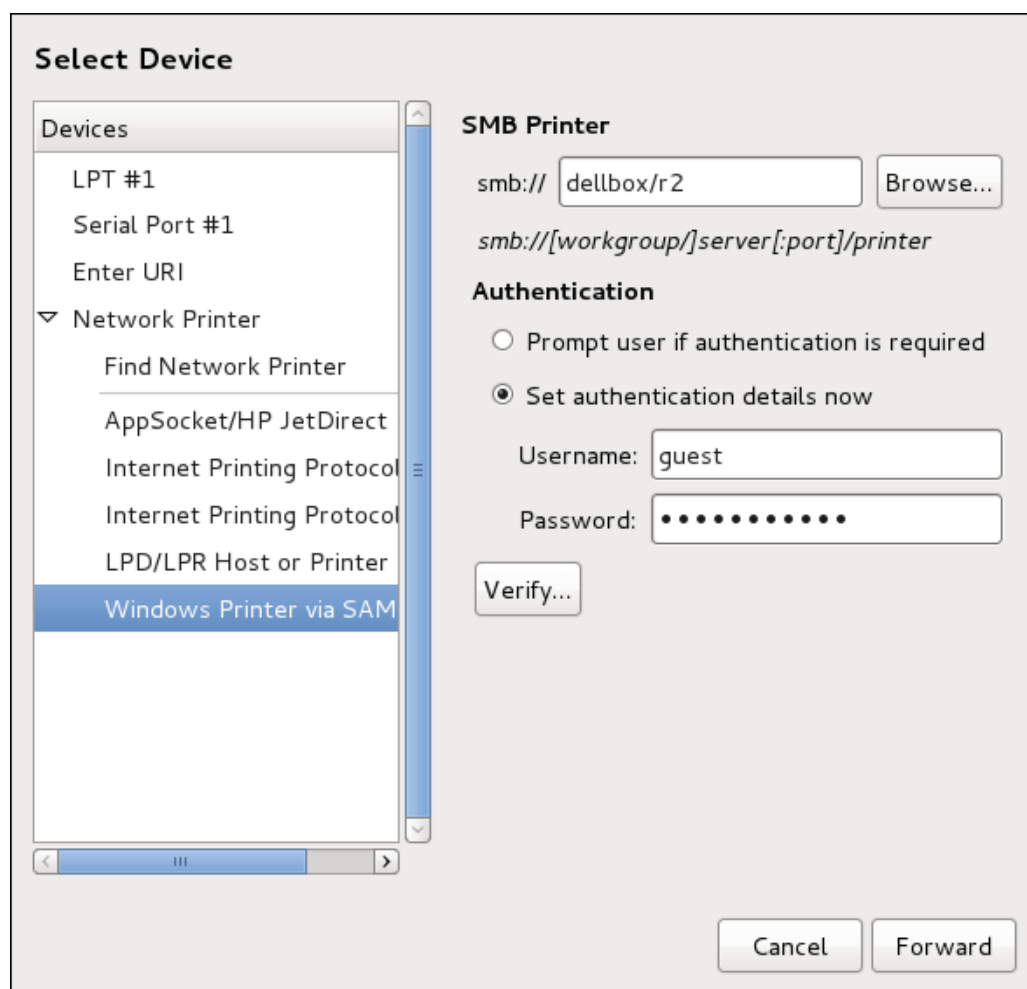


Figure 13.7. Adding a SMB printer

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:
 - Prompt user if authentication is required:** user name and password are collected from the user when printing a document.
 - 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.



Be careful when choosing a password

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 Fedora 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 13.3.8, “Selecting the Printer Model and Finishing”](#) for details.

13.3.8. 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 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.

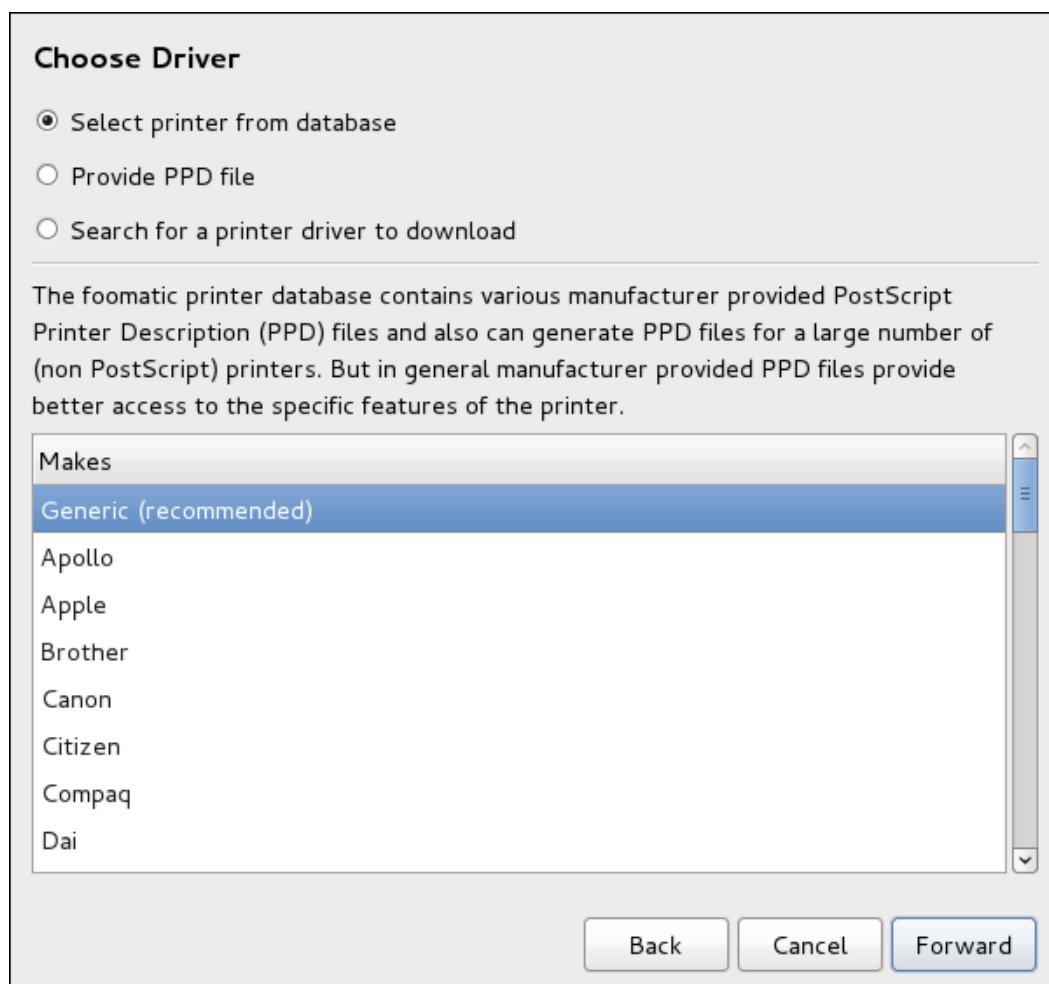


Figure 13.8. Selecting a printer brand

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

Selecting a printer driver

On the right, the recommended 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.

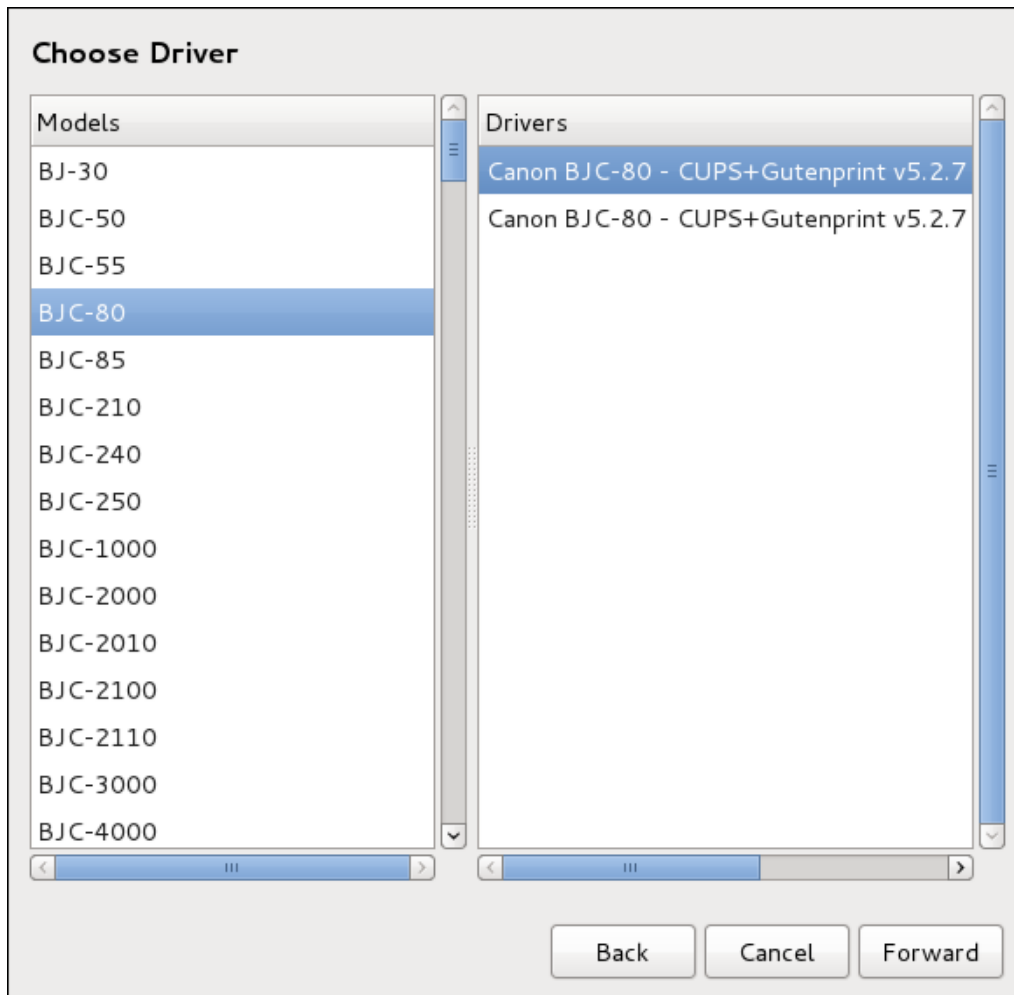


Figure 13.9. Selecting a printer model

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.

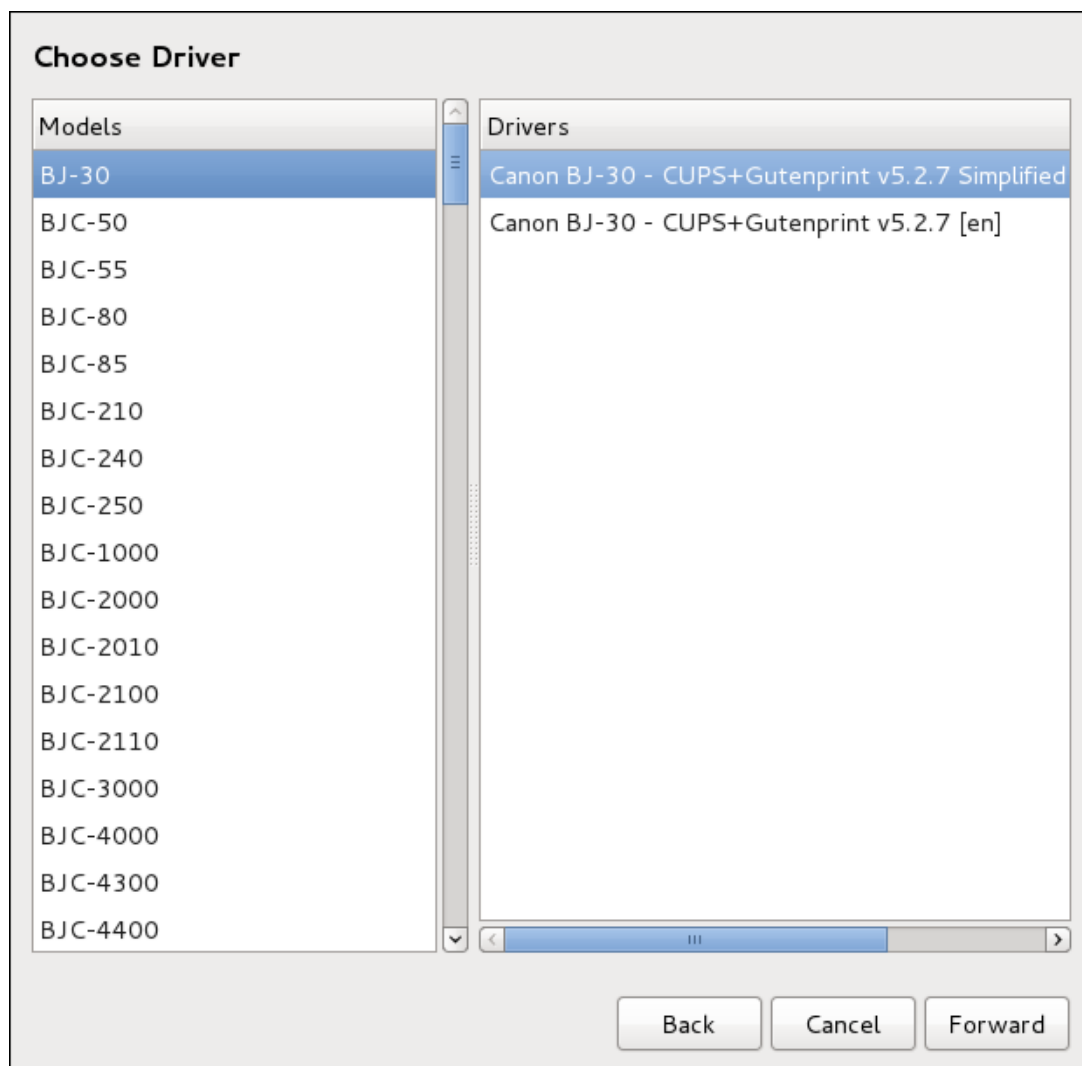


Figure 13.10. Printer setup

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 **Print Test Page** to print a test page now. Alternatively, you can print a test page later as described in [Section 13.3.9, "Printing a Test Page"](#).

13.3.9. 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.

13.3.10. 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.

13.3.10.1. 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.

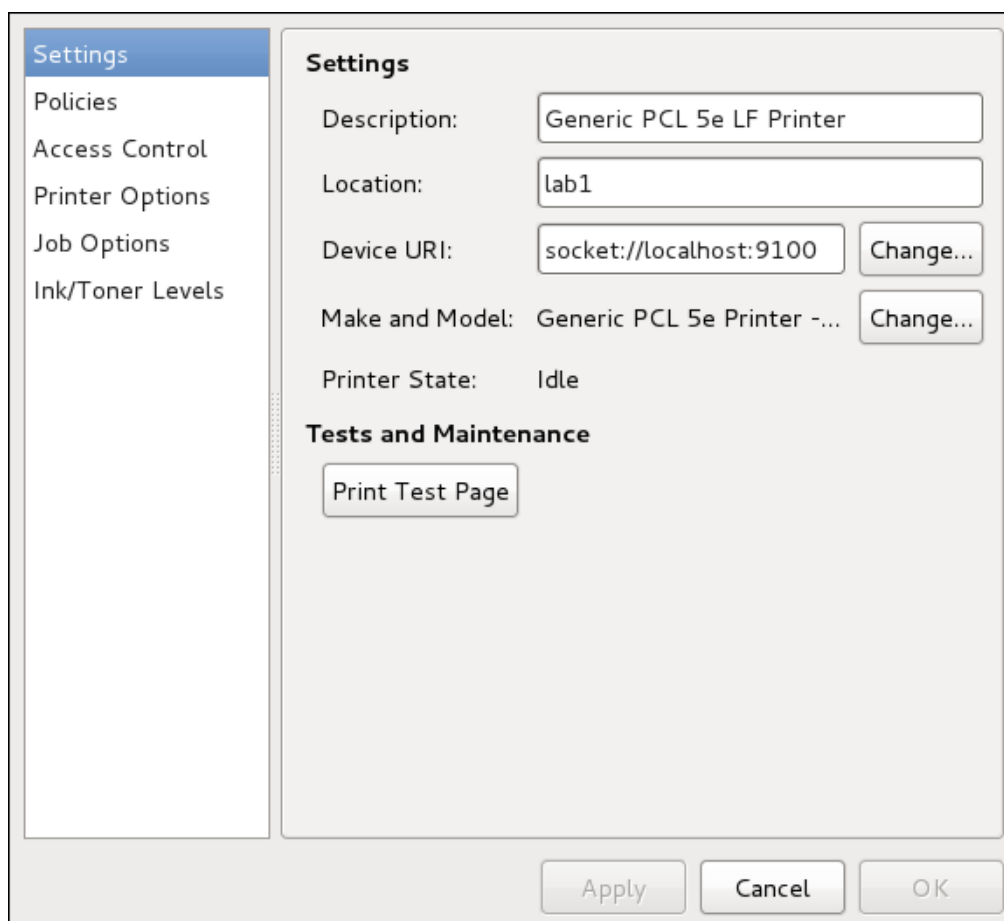


Figure 13.11. Settings page

13.3.10.2. 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 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**).

13.3.10.2.1. 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**.

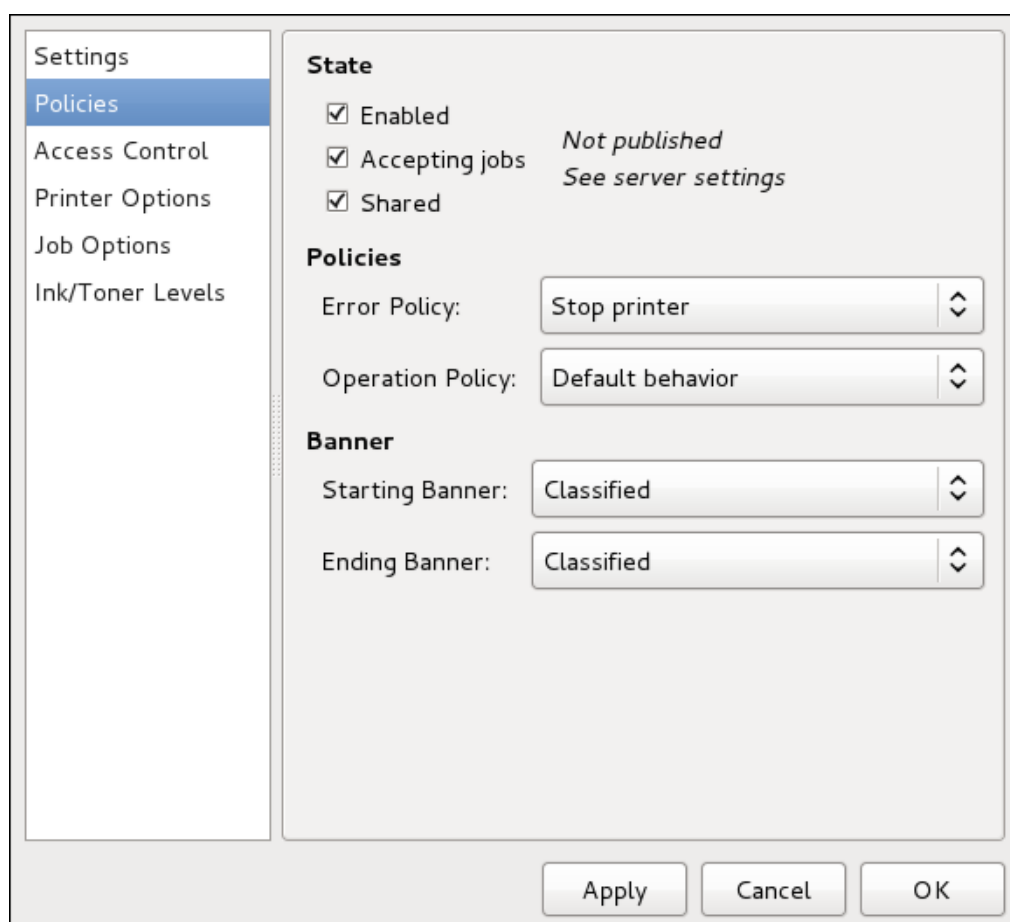


Figure 13.12. Policies page

Make sure that the firewall allows incoming TCP connections to port **631**, the port for the Network Printing Server (IPP) protocol. To allow IPP traffic through the firewall on Fedora 26, make use of `firewalld`'s IPP service. To do so, proceed as follows:

Procedure 13.5. Enabling IPP Service in `firewalld`

1. To start the graphical **firewall-config** tool, press the **Super** key to enter the Activities Overview, type **firewall** and then press **Enter**. The **Firewall Configuration** window opens. You will be prompted for an administrator or root password.

Alternatively, to start the graphical firewall configuration tool using the command line, enter the following command as root user:

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

The **Firewall Configuration** window opens.

Look for the word “Connected” in the lower left corner. This indicates that the **firewall-config** tool is connected to the user space daemon, **firewalld**.

To immediately change the current firewall settings, ensure the drop-down selection menu labeled **Configuration** is set to **Runtime**. Alternatively, to edit the settings to be applied at the next system start, or firewall reload, select **Permanent** from the drop-down list.

2. Select the **Zones** tab and then select the firewall zone to correspond with the network interface to be used. The default is the **public** zone. The **Interfaces** tab shows what interfaces have been assigned to a zone.
3. Select the **Services** tab and then select the **ipp** service to enable sharing. The **ipp-client** service is required for accessing network printers.
4. Close the **firewall-config** tool.

13.3.10.2.2. 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.

Settings
Policies
Access Control
Printer Options
Job Options
Ink/Toner Levels

☐ Allow printing for everyone except these users:
☒ Deny printing for everyone except these users:

Jane Add

Users Delete

John

Apply Cancel OK

Figure 13.13. Access Control page

13.3.10.2.3. The Printer Options Page

The **Printer Options** page contains various configuration options for the printer media and output, and its content may vary from printer to printer. It contains general printing, paper, quality, and printing size settings.

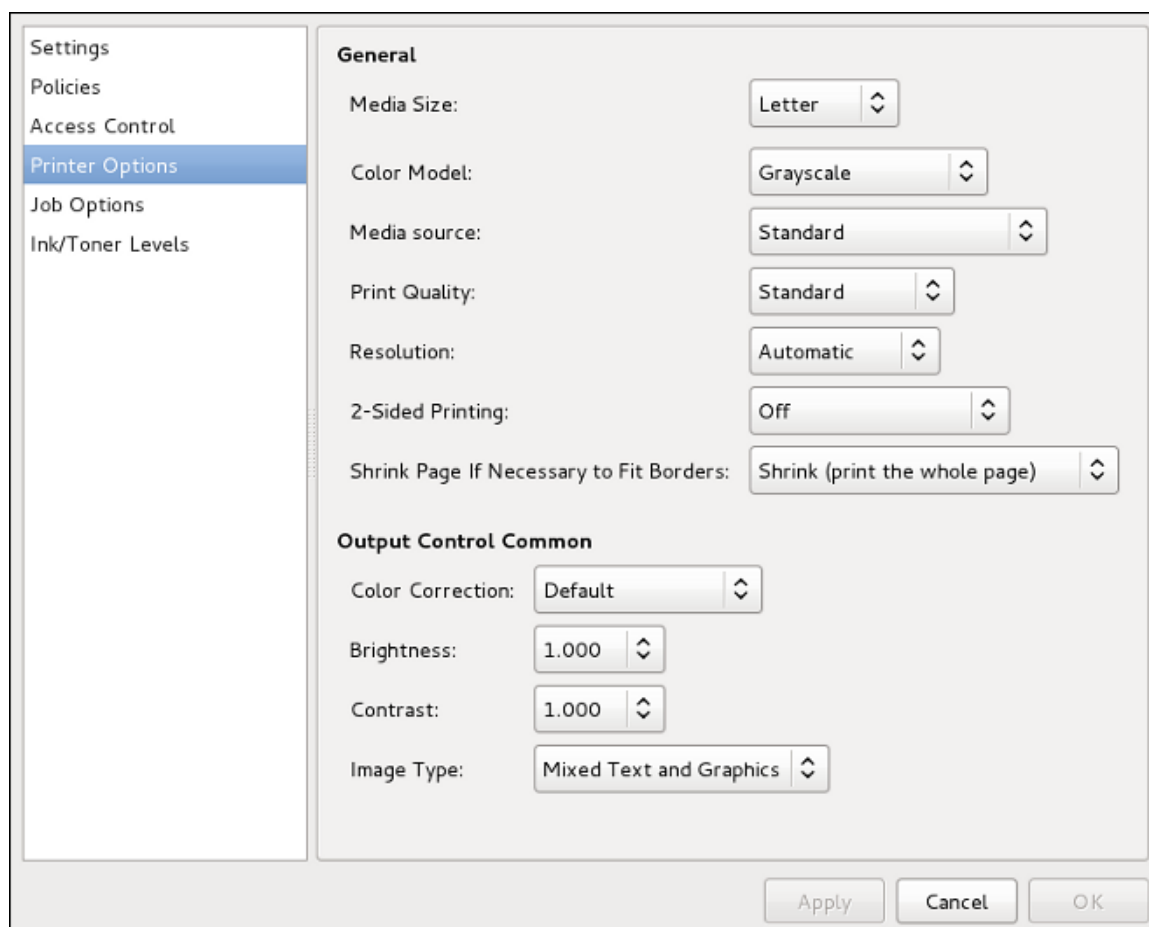


Figure 13.14. Printer Options page

13.3.10.2.4. 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.

Settings
Policies
Access Control
Printer Options
Job Options
Ink/Toner Levels

Specify the default job options for this printer.
Jobs arriving at this print server will have these options added if they are not already set by the application.

Common Options

Copies: 1

Orientation: Automatic rotation

☐ Scale to fit

Pages per side: 1

▶ More

Image Options

☐ Mirror

Scaling: 100 %

▶ More

Text Options

Characters per inch: 10.00

Lines per inch: 6.00

Figure 13.15. Job Options page

13.3.10.2.5. 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.

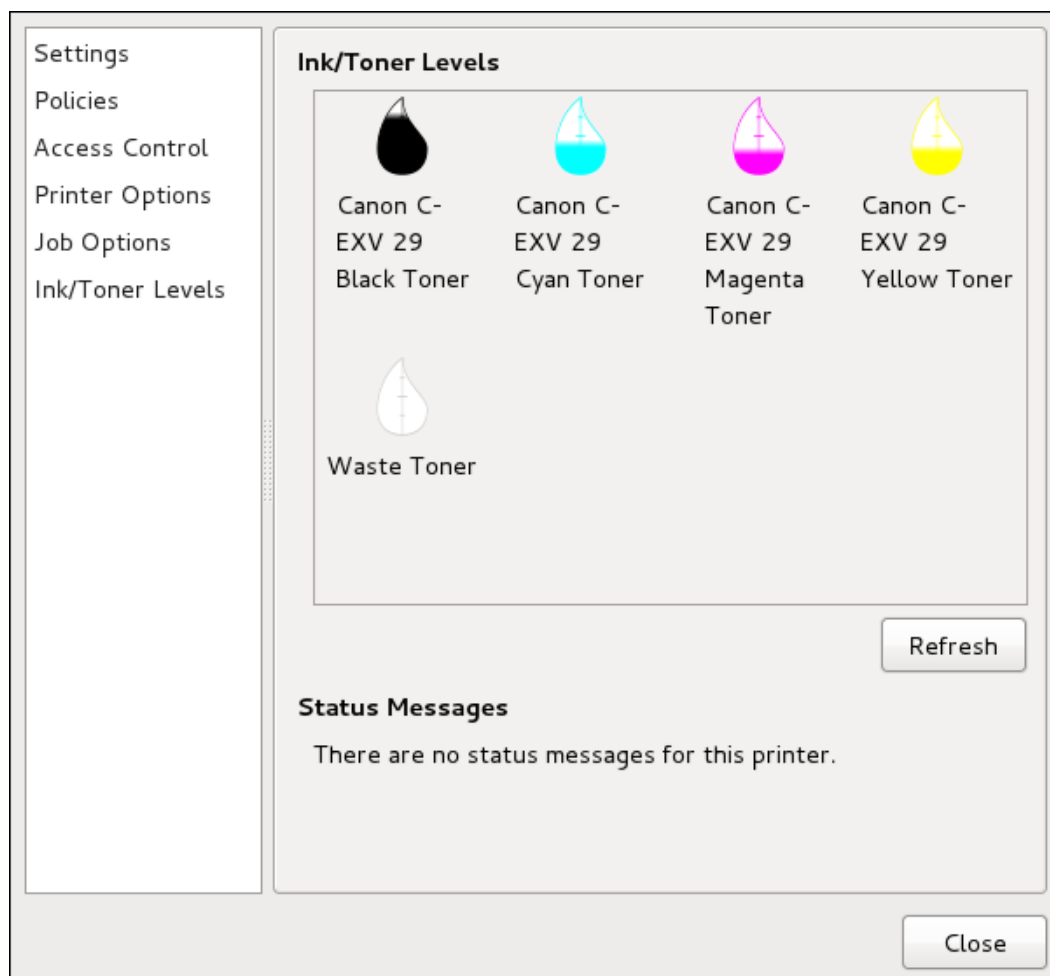


Figure 13.16. Ink/Toner Levels page

13.3.10.3. 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, messages informing about the process appear in the notification area.

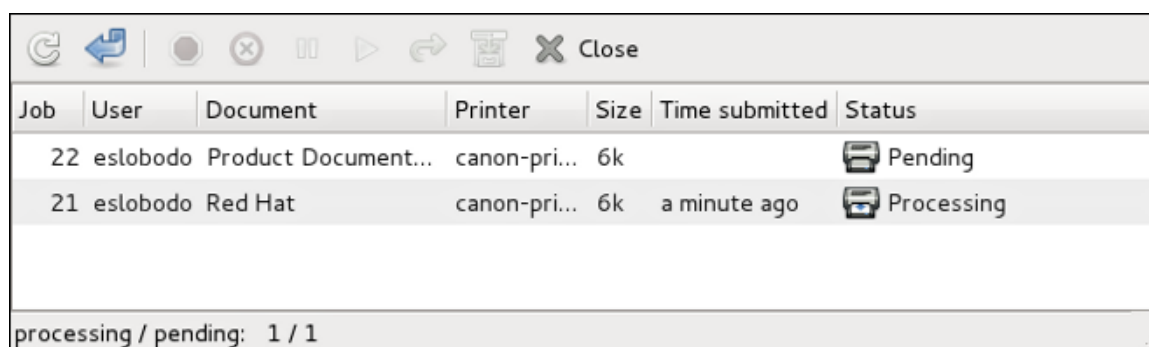


Figure 13.17. GNOME Print Status

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 13.10. 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 13.10](#), “*Example of `lpstat -o` output*”. You cannot 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.

13.3.11. Additional Resources

To learn more about printing on Fedora, see the following resources.

13.3.11.1. Installed Documentation

man lp

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

man cancel

The manual page for the command-line utility to remove print jobs from the print queue.

man mpage

The manual page for the command-line utility to print multiple pages on one sheet of paper.

man cupsd

The manual page for the CUPS printer daemon.

man cupsd.conf

The manual page for the CUPS printer daemon configuration file.

man classes.conf

The manual page for the class configuration file for CUPS.

man lpstat

The manual page for the **lpstat** command, which displays status information about classes, jobs, and printers.

13.3.11.2. 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.

Configuring NTP Using the chrony Suite

Accurate time keeping is important for a number of reasons in IT. In networking for example, accurate time stamps in packets and logs are required. In Linux systems, the NTP protocol is implemented by a daemon running in user space.

The user space daemon updates the system clock running in the kernel. The system clock can keep time by using various clock sources. Usually, the *Time Stamp Counter* (TSC) is used. The TSC is a CPU register which counts the number of cycles since it was last reset. It is very fast, has a high resolution, and there are no interrupts.

There is a choice between the daemons `ntpd` and `chronyd`, which are available from the repositories in the `ntp` and `chrony` packages respectively. This section describes the use of the **chrony** suite of utilities to update the system clock on systems that do not fit into the conventional permanently networked, always on, dedicated server category.

14.1. Introduction to the chrony Suite

Chrony consists of `chronyd`, a daemon that runs in user space, and **chronyc**, a command line program for making adjustments to `chronyd`. Systems which are not permanently connected, or not permanently powered up, take a relatively long time to adjust their system clocks with `ntpd`. This is because many small corrections are made based on observations of the clocks drift and offset. Temperature changes, which may be significant when powering up a system, affect the stability of hardware clocks. Although adjustments begin within a few milliseconds of booting a system, acceptable accuracy may take anything from ten seconds from a warm restart to a number of hours depending on your requirements, operating environment and hardware. **chrony** is a different implementation of the NTP protocol than `ntpd`, it can adjust the system clock more rapidly.

14.1.1. Differences Between `ntpd` and `chronyd`

One of the main differences between `ntpd` and `chronyd` is in the algorithms used to control the computer's clock. Things `chronyd` can do better than `ntpd` are:

- `chronyd` can work well when external time references are only intermittently accessible, whereas `ntpd` needs regular polling of time reference to work well.
- `chronyd` can perform well even when the network is congested for longer periods of time.
- `chronyd` can usually synchronize the clock faster and with better time accuracy.
- `chronyd` quickly adapts to sudden changes in the rate of the clock, for example, due to changes in the temperature of the crystal oscillator, whereas `ntpd` may need a long time to settle down again.
- In the default configuration, `chronyd` never steps the time after the clock has been synchronized at system start, in order not to upset other running programs. `ntpd` can be configured to never step the time too, but it has to use a different means of adjusting the clock, which has some disadvantages.
- `chronyd` can adjust the rate of the clock on a Linux system in a larger range, which allows it to operate even on machines with a broken or unstable clock. For example, on some virtual machines.

Things `chronyd` can do that `ntpd` cannot do:

- `chronyd` provides support for isolated networks where the only method of time correction is manual entry. For example, by the administrator looking at a clock. `chronyd` can examine the errors corrected at different updates to estimate the rate at which the computer gains or loses time, and use this estimate to trim the computer clock subsequently.
- `chronyd` provides support to work out the rate of gain or loss of the real-time clock, the hardware clock, that maintains the time when the computer is turned off. It can use this data when the system boots to set the system time using an adjusted value of the time taken from the real-time clock. This is, at time of writing, only available in Linux.

Things `ntpd` can do that `chronyd` cannot do:

- `ntpd` fully supports NTP version 4 (*RFC 5905*), including broadcast, multicast, manycast clients and servers, and the orphan mode. It also supports extra authentication schemes based on public-key cryptography (*RFC 5906*). `chronyd` uses NTP version 3 (*RFC 1305*), which is compatible with version 4.
- `ntpd` includes drivers for many reference clocks whereas `chronyd` relies on other programs, for example `gpsd`, to access the data from the reference clocks.

14.1.2. Choosing Between NTP Daemons

- **Chrony** should be considered for all systems which are frequently suspended or otherwise intermittently disconnected and reconnected to a network. Mobile and virtual systems for example.
- The NTP daemon (`ntpd`) should be considered for systems which are normally kept permanently on. Systems which are required to use broadcast or multicast IP, or to perform authentication of packets with the Autokey protocol, should consider using `ntpd`. **Chrony** only supports symmetric key authentication using a message authentication code (MAC) with MD5, SHA1 or stronger hash functions, whereas `ntpd` also supports the Autokey authentication protocol which can make use of the PKI system. Autokey is described in *RFC 5906*.

14.2. Understanding chrony and Its Configuration

14.2.1. Understanding chronyd

The **chrony** daemon, `chronyd`, running in user space, makes adjustments to the system clock which is running in the kernel. It does this by consulting external time sources, using the NTP protocol, when ever network access allows it to do so. When external references are not available, `chronyd` will use the last calculated drift stored in the drift file. It can also be commanded manually to make corrections, by **chronyc**.

14.2.2. Understanding chronyc

The **chrony** daemon, `chronyd`, can be controlled by the command line utility **chronyc**. This utility provides a command prompt which allows entering of a number of commands to make changes to `chronyd`. The default configuration is for `chronyd` to only accept commands from a local instance of **chronyc**, but **chronyc** can be used to alter the configuration so that `chronyd` will allow external control. **chronyc** can be run remotely after first configuring `chronyd` to accept remote connections. The IP addresses allowed to connect to `chronyd` should be tightly controlled.

14.2.3. Understanding the chrony Configuration Commands

The default configuration file for chronyd is `/etc/chrony.conf`. The `-f` option can be used to specify an alternate configuration file path. See the chronyd man page for further options. For a complete list of the directives that can be used see <http://chrony.tuxfamily.org/manual.html#Configuration-file>. Below is a selection of configuration options:

Comments

Comments should be preceded by `#`, `%`, `;` or `!`

allow

Optionally specify a host, subnet, or network from which to allow NTP connections to a machine acting as NTP server. The default is not to allow connections.

Examples:

1. `allow server1.example.com`

Use this form to specify a particular host, by its host name, to be allowed access.

2. `allow 192.0.2.0/24`

Use this form to specify a particular network to be allowed access.

3. `allow 2001:db8::/32`

Use this form to specify an IPv6 address to be allowed access.

cmdallow

This is similar to the **allow** directive (see section **allow**), except that it allows control access (rather than NTP client access) to a particular subnet or host. (By “control access” is meant that **chronyc** can be run on those hosts and successfully connect to chronyd on this computer.) The syntax is identical. There is also a **cmddeny all** directive with similar behavior to the **cmdallow all** directive.

dumpdir

Path to the directory to save the measurement history across restarts of chronyd (assuming no changes are made to the system clock behavior whilst it is not running). If this capability is to be used (via the **dumponexit** command in the configuration file, or the **dump** command in **chronyc**), the **dumpdir** command should be used to define the directory where the measurement histories are saved.

dumponexit

If this command is present, it indicates that chronyd should save the measurement history for each of its time sources recorded whenever the program exits. (See the **dumpdir** command above).

local

The **local** keyword is used to allow chronyd to appear synchronized to real time from the viewpoint of clients polling it, even if it has no current synchronization source. This option is normally used on the “master” computer in an isolated network, where several computers are required to synchronize to one another, and the “master” is kept in line with real time by manual input.

An example of the command is:


```
local stratum 10
```

A large value of 10 indicates that the clock is so many hops away from a reference clock that its time is unreliable. If the computer ever has access to another computer which is ultimately synchronized to a reference clock, it will almost certainly be at a stratum less than 10. Therefore, the choice of a high value like 10 for the **local** command prevents the machine's own time from ever being confused with real time, were it ever to leak out to clients that have visibility of real servers.

log

The **log** command indicates that certain information is to be logged. It accepts the following options:

measurements

This option logs the raw NTP measurements and related information to a file called **measurements.log**.

statistics

This option logs information about the regression processing to a file called **statistics.log**.

tracking

This option logs changes to the estimate of the system's gain or loss rate, and any slews made, to a file called **tracking.log**.

rtc

This option logs information about the system's real-time clock.

refclocks

This option logs the raw and filtered reference clock measurements to a file called **refclocks.log**.

tempcomp

This option logs the temperature measurements and system rate compensations to a file called **tempcomp.log**.

The log files are written to the directory specified by the **logdir** command. An example of the command is:

```
log measurements statistics tracking
```

logdir

This directive allows the directory where log files are written to be specified. An example of the use of this directive is:

```
logdir /var/log/chrony
```

makestep

Normally chronyd will cause the system to gradually correct any time offset, by slowing down or speeding up the clock as required. In certain situations, the system clock may be so far adrift that this slewing process would take a very long time to correct the system clock. This directive forces chronyd to step system clock if the adjustment is larger than a threshold value, but only if there were no more clock updates since chronyd was started than a specified limit (a negative value can be used to disable the limit). This is particularly useful when using reference clocks, because the **initstepslew** directive only works with NTP sources.

An example of the use of this directive is:

```
makestep 1000 10
```

This would step the system clock if the adjustment is larger than 1000 seconds, but only in the first ten clock updates.

maxchange

This directive sets the maximum allowed offset corrected on a clock update. The check is performed only after the specified number of updates to allow a large initial adjustment of the system clock. When an offset larger than the specified maximum occurs, it will be ignored for the specified number of times and then `chronyd` will give up and exit (a negative value can be used to never exit). In both cases a message is sent to `syslog`.

An example of the use of this directive is:

```
maxchange 1000 1 2
```

After the first clock update, `chronyd` will check the offset on every clock update, it will ignore two adjustments larger than 1000 seconds and exit on another one.

maxupdateskew

One of `chronyd`'s tasks is to work out how fast or slow the computer's clock runs relative to its reference sources. In addition, it computes an estimate of the error bounds around the estimated value. If the range of error is too large, it indicates that the measurements have not settled down yet, and that the estimated gain or loss rate is not very reliable. The **maxupdateskew** parameter is the threshold for determining whether an estimate is too unreliable to be used. By default, the threshold is 1000 ppm. The format of the syntax is:

```
maxupdateskew skew-in-ppm
```

Typical values for *skew-in-ppm* might be 100 for a dial-up connection to servers over a telephone line, and 5 or 10 for a computer on a LAN. It should be noted that this is not the only means of protection against using unreliable estimates. At all times, `chronyd` keeps track of both the estimated gain or loss rate, and the error bound on the estimate. When a new estimate is generated following another measurement from one of the sources, a weighted combination algorithm is used to update the master estimate. So if `chronyd` has an existing highly-reliable master estimate and a new estimate is generated which has large error bounds, the existing master estimate will dominate in the new master estimate.

noclientlog

This directive, which takes no arguments, specifies that client accesses are not to be logged.

Normally they are logged, allowing statistics to be reported using the `clients` command in **chronyc**.

reselectdist

When `chronyd` selects synchronization source from available sources, it will prefer the one with minimum synchronization distance. However, to avoid frequent reselecting when there are sources with similar distance, a fixed distance is added to the distance for sources that are currently not selected. This can be set with the **reselectdist** option. By default, the distance is 100 microseconds.

The format of the syntax is:

```
reselectdist dist-in-seconds
```


stratumweight

The **stratumweight** directive sets how much distance should be added per stratum to the synchronization distance when chronyd selects the synchronization source from available sources.

The format of the syntax is:

```
stratumweight dist-in-seconds
```

By default, *dist-in-seconds* is 1 second. This means that sources with lower stratum are usually preferred to sources with higher stratum even when their distance is significantly worse. Setting **stratumweight** to 0 makes chronyd ignore stratum when selecting the source.

rtcfile

The **rtcfile** directive defines the name of the file in which chronyd can save parameters associated with tracking the accuracy of the system's real-time clock (RTC). The format of the syntax is:

```
rtcfile /var/lib/chrony/rtc
```

chronyd saves information in this file when it exits and when the **writertc** command is issued in **chronyc**. The information saved is the RTC's error at some epoch, that epoch (in seconds since January 1 1970), and the rate at which the RTC gains or loses time. Not all real-time clocks are supported as their code is system-specific. Note that if this directive is used then the real-time clock should not be manually adjusted as this would interfere with **chrony**'s need to measure the rate at which the real-time clock drifts if it was adjusted at random intervals.

rtcsync

The **rtcsync** directive is present in the **/etc/chrony.conf** file by default. This will inform the kernel the system clock is kept synchronized and the kernel will update the real-time clock every 11 minutes.

14.2.4. Security with chronyc

As access to **chronyc** allows changing chronyd just as editing the configuration files would, access to **chronyc** should be limited. Passwords can be specified in the key file, written in ASCII or HEX, to restrict the use of **chronyc**. One of the entries is used to restrict the use of operational commands and is referred to as the command key. In the default configuration, a random command key is generated automatically on start. It should not be necessary to specify or alter it manually.

Other entries in the key file can be used as NTP keys to authenticate packets received from remote NTP servers or peers. The two sides need to share a key with identical ID, hash type and password in their key file. This requires manually creating the keys and copying them over a secure medium, such as SSH. If the key ID was, for example, 10 then the systems that act as clients must have a line in their configuration files in the following format:

```
server w.x.y.z key 10
peer w.x.y.z key 10
```

The location of the key file is specified in the **/etc/chrony.conf** file. The default entry in the configuration file is:

```
keyfile /etc/chrony.keys
```


The command key number is specified in `/etc/chrony.conf` using the **commandkey** directive, it is the key chronyd will use for authentication of user commands. The directive in the configuration file takes the following form:

```
commandkey 1
```

An example of the format of the default entry in the key file, `/etc/chrony.keys`, for the command key is:

```
1 SHA1 HEX:A6CFC50C9C93AB6E5A19754C246242FC5471BCDF
```

Where **1** is the key ID, SHA1 is the hash function to use, **HEX** is the format of the key, and **A6CFC50C9C93AB6E5A19754C246242FC5471BCDF** is the key randomly generated when **chronyd** was started for the first time. The key can be given in hexadecimal or ASCII format (the default).

A manual entry in the key file, used to authenticate packets from certain NTP servers or peers, can be as simple as the following:

```
20 foobar
```

Where **20** is the key ID and **foobar** is the secret authentication key. The default hash is MD5, and ASCII is the default format for the key.

By default, chronyd is configured to listen for commands only from localhost (127.0.0.1 and ::1) on port **323**. To access chronyd remotely with **chronyc**, any **bindcmdaddress** directives in the `/etc/chrony.conf` file should be removed to enable listening on all interfaces and the **cmdallow** directive should be used to allow commands from the remote IP address, network, or subnet. In addition, port **323** has to be opened in the firewall in order to connect from a remote system. Note that the **allow** directive is for NTP access whereas the **cmdallow** directive is to enable the receiving of remote commands. It is possible to make these changes temporarily using **chronyc** running locally. Edit the configuration file to make persistent changes.

The communication between **chronyc** and **chronyd** is done over UDP, so it needs to be authorized before issuing operational commands. To authorize, use the **authhash** and **password** commands as follows:

```
chronyc> authhash SHA1
chronyc> password HEX:A6CFC50C9C93AB6E5A19754C246242FC5471BCDF
200 OK
```

If **chronyc** is used to configure the local **chronyd**, the **-a** option will run the **authhash** and **password** commands automatically.

Only the following commands can be used without providing a password: **activity**, **authhash**, **dns**, **exit**, **help**, **password**, **quit**, **rtcddata**, **sources**, **sourcestats**, **tracking**, **waitsync**.

14.3. Using chrony

14.3.1. Installing chrony

The **chrony** suite is installed by default on some versions of Fedora. If required, to ensure that it is, run the following command as root:


```
~]# dnf install chrony
```

The default location for the **chrony** daemon is **/usr/sbin/chronyd**. The command line utility will be installed to **/usr/bin/chronyc**.

14.3.2. Checking the Status of chronyd

To check the status of chronyd, issue the following command:

```
~]$ systemctl status chronyd
chronyd.service - NTP client/server
   Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled)
   Active: active (running) since Wed 2013-06-12 22:23:16 CEST; 11h ago
```

14.3.3. Starting chronyd

To start chronyd, issue the following command as root:

```
~]# systemctl start chronyd
```

To ensure chronyd starts automatically at system start, issue the following command as root:

```
~]# systemctl enable chronyd
```

14.3.4. Stopping chronyd

To stop chronyd, issue the following command as root:

```
~]# systemctl stop chronyd
```

To prevent chronyd from starting automatically at system start, issue the following command as root:

```
~]# systemctl disable chronyd
```

14.3.5. Checking if chrony is Synchronized

To check if **chrony** is synchronized, make use of the **tracking**, **sources**, and **sourcestats** commands.

14.3.5.1. Checking chrony Tracking

To check **chrony** tracking, issue the following command:

```
~]$ chronyc tracking
Reference ID    : 1.2.3.4 (a.b.c)
Stratum        : 3
Ref time (UTC) : Fri Feb  3 15:00:29 2012
System time    : 0.000001501 seconds slow of NTP time
Last offset    : -0.000001632 seconds
RMS offset     : 0.000002360 seconds
Frequency      : 331.898 ppm fast
Residual freq  : 0.004 ppm
Skew           : 0.154 ppm
```



```

Root delay      : 0.373169 seconds
Root dispersion : 0.024780 seconds
Update interval : 64.2 seconds
Leap status     : Normal

```

The fields are as follows:

Reference ID

This is the reference ID and name (or IP address) if available, of the server to which the computer is currently synchronized. If this is 127.127.1.1 it means the computer is not synchronized to any external source and that you have the “local” mode operating (via the local command in **chronyc**, or the **local** directive in the **/etc/chrony.conf** file (see section **local**)).

Stratum

The stratum indicates how many hops away from a computer with an attached reference clock we are. Such a computer is a stratum-1 computer, so the computer in the example is two hops away (that is to say, a.b.c is a stratum-2 and is synchronized from a stratum-1).

Ref time

This is the time (UTC) at which the last measurement from the reference source was processed.

System time

In normal operation, chronyd never steps the system clock, because any jump in the timescale can have adverse consequences for certain application programs. Instead, any error in the system clock is corrected by slightly speeding up or slowing down the system clock until the error has been removed, and then returning to the system clock's normal speed. A consequence of this is that there will be a period when the system clock (as read by other programs using the `gettimeofday()` system call, or by the `date` command in the shell) will be different from chronyd's estimate of the current true time (which it reports to NTP clients when it is operating in server mode). The value reported on this line is the difference due to this effect.

Last offset

This is the estimated local offset on the last clock update.

RMS offset

This is a long-term average of the offset value.

Frequency

The “frequency” is the rate by which the system's clock would be wrong if chronyd was not correcting it. It is expressed in ppm (parts per million). For example, a value of 1ppm would mean that when the system's clock thinks it has advanced 1 second, it has actually advanced by 1.000001 seconds relative to true time.

Residual freq

This shows the “residual frequency” for the currently selected reference source. This reflects any difference between what the measurements from the reference source indicate the frequency should be and the frequency currently being used. The reason this is not always zero is that a smoothing procedure is applied to the frequency. Each time a measurement from the reference source is obtained and a new residual frequency computed, the estimated accuracy of this residual is compared with the estimated accuracy (see **skew** next) of the existing frequency value. A weighted average is computed for the new frequency, with weights depending on these accuracies. If the measurements from the reference source follow a consistent trend, the residual will be driven to zero over time.

Skew

This is the estimated error bound on the frequency.

Root delay

This is the total of the network path delays to the stratum-1 computer from which the computer is ultimately synchronized. In certain extreme situations, this value can be negative. (This can arise in a symmetric peer arrangement where the computers' frequencies are not tracking each other and the network delay is very short relative to the turn-around time at each computer.)

Root dispersion

This is the total dispersion accumulated through all the computers back to the stratum-1 computer from which the computer is ultimately synchronized. Dispersion is due to system clock resolution, statistical measurement variations etc.

Leap status

This is the leap status, which can be Normal, Insert second, Delete second or Not synchronized.

14.3.5.2. Checking chrony Sources

The sources command displays information about the current time sources that chronyd is accessing. The optional argument -v can be specified, meaning verbose. In this case, extra caption lines are shown as a reminder of the meanings of the columns.

```
~]$ chronyc sources
210 Number of sources = 3
MS Name/IP address         Stratum Poll Reach LastRx Last sample
=====
#* GPS0                     0    4   377    11  -479ns[ -621ns] +/-  134ns
^? a.b.c                    2    6   377    23  -923us[ -924us] +/-   43ms
^+ d.e.f                    1    6   377    21 -2629us[ -2619us] +/-   86ms
```

The columns are as follows:

M

This indicates the mode of the source. ^ means a server, = means a peer and # indicates a locally connected reference clock.

S

This column indicates the state of the sources. "*" indicates the source to which chronyd is currently synchronized. "+" indicates acceptable sources which are combined with the selected source. "-" indicates acceptable sources which are excluded by the combining algorithm. "?" indicates sources to which connectivity has been lost or whose packets do not pass all tests. "x" indicates a clock which chronyd thinks is a *false ticker* (its time is inconsistent with a majority of other sources). "~" indicates a source whose time appears to have too much variability. The "?" condition is also shown at start-up, until at least 3 samples have been gathered from it.

Name/IP address

This shows the name or the IP address of the source, or reference ID for reference clocks.

Stratum

This shows the stratum of the source, as reported in its most recently received sample. Stratum 1 indicates a computer with a locally attached reference clock. A computer that is synchronized to a stratum 1 computer is at stratum 2. A computer that is synchronized to a stratum 2 computer is at stratum 3, and so on.

Poll

This shows the rate at which the source is being polled, as a base-2 logarithm of the interval in seconds. Thus, a value of 6 would indicate that a measurement is being made every 64 seconds. chronyd automatically varies the polling rate in response to prevailing conditions.

Reach

This shows the source's reach register printed as an octal number. The register has 8 bits and is updated on every received or missed packet from the source. A value of 377 indicates that a valid reply was received for all of the last eight transmissions.

LastRx

This column shows how long ago the last sample was received from the source. This is normally in seconds. The letters **m**, **h**, **d** or **y** indicate minutes, hours, days or years. A value of 10 years indicates there were no samples received from this source yet.

Last sample

This column shows the offset between the local clock and the source at the last measurement. The number in the square brackets shows the actual measured offset. This may be suffixed by **ns** (indicating nanoseconds), **us** (indicating microseconds), **ms** (indicating milliseconds), or **s** (indicating seconds). The number to the left of the square brackets shows the original measurement, adjusted to allow for any slews applied to the local clock since. The number following the **+/-** indicator shows the margin of error in the measurement. Positive offsets indicate that the local clock is ahead of the source.

14.3.5.3. Checking chrony Source Statistics

The **sourcestats** command displays information about the drift rate and offset estimation process for each of the sources currently being examined by chronyd. The optional argument **-v** can be specified, meaning verbose. In this case, extra caption lines are shown as a reminder of the meanings of the columns.

```
~]$ chronyc sourcestats
210 Number of sources = 1
Name/IP Address      NP  NR  Span  Frequency  Freq Skew  Offset  Std Dev
=====
abc.def.ghi          11   5  46m   -0.001     0.045     1us    25us
```

The columns are as follows:

Name/IP address

This is the name or IP address of the NTP server (or peer) or reference ID of the reference clock to which the rest of the line relates.

NP

This is the number of sample points currently being retained for the server. The drift rate and current offset are estimated by performing a linear regression through these points.

NR

This is the number of runs of residuals having the same sign following the last regression. If this number starts to become too small relative to the number of samples, it indicates that a straight line is no longer a good fit to the data. If the number of runs is too low, chronyd discards older samples and re-runs the regression until the number of runs becomes acceptable.

Span

This is the interval between the oldest and newest samples. If no unit is shown the value is in seconds. In the example, the interval is 46 minutes.

Frequency

This is the estimated residual frequency for the server, in parts per million. In this case, the computer's clock is estimated to be running 1 part in 10^9 slow relative to the server.

Freq Skew

This is the estimated error bounds on Freq (again in parts per million).

Offset

This is the estimated offset of the source.

Std Dev

This is the estimated sample standard deviation.

14.3.6. Manually Adjusting the System Clock

To update, or step, the system clock immediately, bypassing any adjustments in progress by slewing the clock, issue the following commands as root:

```
~]# chronyc
chrony> password commandkey-password
200 OK
chrony> makestep
200 OK
```

Where *commandkey-password* is the command key or password stored in the key file.

If the **rtcfile** directive is used, the real-time clock should not be manually adjusted. Random adjustments would interfere with **chrony**'s need to measure the rate at which the real-time clock drifts.

If **chronyc** is used to configure the local **chronyd**, the **-a** will run the **authhash** and **password** commands automatically. This means that the interactive session illustrated above can be replaced by:

```
chronyc -a makestep
```

14.4. Setting Up chrony for Different Environments

14.4.1. Setting Up chrony for a System Which is Infrequently Connected

This example is intended for systems which use dial-on-demand connections. The normal configuration should be sufficient for mobile and virtual devices which connect intermittently. First, review and confirm that the default settings in the **/etc/chrony.conf** are similar to the following:

```
driftfile /var/lib/chrony/drift
commandkey 1
keyfile /etc/chrony.keys
```

The command key ID is generated at install time and should correspond with the **commandkey** value in the key file, **/etc/chrony.keys**.

Using your editor running as root, add the addresses of four NTP servers as follows:

```
server 0.pool.ntp.org offline
server 1.pool.ntp.org offline
server 2.pool.ntp.org offline
server 3.pool.ntp.org offline
```

The **offline** option can be useful in preventing systems from trying to activate connections. The **chrony** daemon will wait for **chronyc** to inform it that the system is connected to the network or Internet.

14.4.2. Setting Up chrony for a System in an Isolated Network

For a network that is never connected to the Internet, one computer is selected to be the master timeserver. The other computers are either direct clients of the master, or clients of clients. On the master, the drift file must be manually set with the average rate of drift of the system clock. If the master is rebooted it will obtain the time from surrounding systems and take an average to set its system clock. Thereafter it resumes applying adjustments based on the drift file. The drift file will be updated automatically when the **settime** command is used.

On the system selected to be the master, using a text editor running as root, edit the **/etc/chrony.conf** as follows:

```
driftfile /var/lib/chrony/drift
commandkey 1
keyfile /etc/chrony.keys
initstepslew 10 client1 client3 client6
local stratum 8
manual
allow 192.0.2.0
```

Where 192.0.2.0 is the network or subnet address from which the clients are allowed to connect.

On the systems selected to be direct clients of the master, using a text editor running as root, edit the **/etc/chrony.conf** as follows:

```
server master
driftfile /var/lib/chrony/drift
logdir /var/log/chrony
log measurements statistics tracking
keyfile /etc/chrony.keys
commandkey 24
local stratum 10
initstepslew 20 master
allow 192.0.2.123
```

Where 192.0.2.123 is the address of the master, and master is the host name of the master. Clients with this configuration will resynchronize the master if it restarts.

On the client systems which are not to be direct clients of the master, the **/etc/chrony.conf** file should be the same except that the **local** and **allow** directives should be omitted.

14.5. Using chronyc

14.5.1. Using chronyc to Control chronyd

To make changes to the local instance of chronyd using the command line utility **chronyc** in interactive mode, enter the following command as root:

```
~]# chronyc -a
```

chronyc must run as root if some of the restricted commands are to be used. The **-a** option is for automatic authentication using the local keys when configuring chronyd on the local system. See [Section 14.2.4, “Security with chronyc”](#) for more information.

The **chronyc** command prompt will be displayed as follows:


```
chronyc>
```

You can type **help** to list all of the commands.

The utility can also be invoked in non-interactive command mode if called together with a command as follows:

```
chronyc command
```

Note

Changes made using **chronyc** are not permanent, they will be lost after a **chronyd** restart. For permanent changes, modify **/etc/chrony.conf**.

14.5.2. Using chronyc for Remote Administration

To configure **chrony** to connect to a remote instance of **chronyd**, issue a command in the following format:

```
~]$ chronyc -h hostname
```

Where *hostname* is the host name to connect to. The default is to connect to the local daemon.

To configure **chrony** to connect to a remote instance of **chronyd** on a non-default port, issue a command in the following format:

```
~]$ chronyc -h hostname -p port
```

Where *port* is the port in use for controlling and monitoring by the remote instance of **chronyd**.

Note that commands issued at the **chronyc** command prompt are not persistent. Only commands in the configuration file are persistent.

The first command must be the **password** command at the **chronyc** command prompt as follows:

```
chronyc> password password
200 OK
```

The password should not have any spaces.

If the password is not an MD5 hash, the hashed password must be preceded by the **authhash** command as follows:

```
chronyc> authhash SHA1
chronyc> password HEX:A6CFC50C9C93AB6E5A19754C246242FC5471BCDF
200 OK
```

The password or hash associated with the command key for a remote system is best obtained by SSH. An SSH connection should be established to the remote machine and the ID of the command key from **/etc/chrony.conf** and the command key in **/etc/chrony.keys** memorized or stored securely for the duration of the session.

14.6. Additional Resources

The following sources of information provide additional resources regarding **chrony**.

14.6.1. Installed Documentation

- **chrony(1)** man page — Introduces the **chrony** daemon and the command-line interface tool.
- **chronyc(1)** man page — Describes the **chronyc** command-line interface tool including commands and command options.
- **chronyd(1)** man page — Describes the **chronyd** daemon including commands and command options.
- **chrony.conf(5)** man page — Describes the **chrony** configuration file.
- **/usr/share/doc/chrony/chrony.txt** — User guide for the **chrony** suite.

14.6.2. Online Documentation

<http://chrony.tuxfamily.org/manual.html>

The online user guide for **chrony**.

Configuring NTP Using ntpd

15.1. 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.

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 time stamps made on different systems must be made by synchronized clocks to be of real value. As systems and networks become increasingly faster, there is a corresponding need for clocks with greater accuracy and resolution. In some countries there are legal obligations to keep accurately synchronized clocks. Please see www.ntp.org for more information. In Linux systems, NTP is implemented by a daemon running in user space. The default NTP user space daemon in Fedora 26 is `chronyd`. It must be disabled if you want to use the `ntpd` daemon. See [Chapter 14, Configuring NTP Using the chrony Suite](#) for information on **chrony**.

The user space daemon updates the system clock, which is a software clock running in the kernel. Linux uses a software clock as its system clock for better resolution than the typical embedded hardware clock referred to as the “*Real Time Clock*” (RTC). See the `rtc(4)` and `hwclock(8)` man pages for information on hardware clocks. The system clock can keep time by using various clock sources. Usually, the *Time Stamp Counter* (TSC) is used. The TSC is a CPU register which counts the number of cycles since it was last reset. It is very fast, has a high resolution, and there are no interrupts. On system start, the system clock reads the time and date from the RTC. The time kept by the RTC will drift away from actual time by up to 5 minutes per month due to temperature variations. Hence the need for the system clock to be constantly synchronized with external time references. When the system clock is being synchronized by `ntpd`, the kernel will in turn update the RTC every 11 minutes automatically.

15.2. 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.

15.3. Understanding NTP

The version of NTP used by Fedora is as described in [RFC 1305 Network Time Protocol \(Version 3\) Specification, Implementation and Analysis](http://www.rfc-editor.org/info/rfc1305)¹ and [RFC 5905 Network Time Protocol Version 4: Protocol and Algorithms Specification](http://www.rfc-editor.org/info/rfc5905)²

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

¹ <http://www.rfc-editor.org/info/rfc1305>

² <http://www.rfc-editor.org/info/rfc5905>

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 3, Configuring the Date and Time](#). With the `-g` option to the `ntpd` command (used by default), any offset at system start will be corrected, but during normal operation only offsets of up to 1000 seconds will be corrected.

Some software may fail or produce an error if the time is changed backwards. For systems that are sensitive to step changes in the time, the threshold can be changed to 600s instead of 128ms using the `-x` option (unrelated to the `-g` option). Using the `-x` option to increase the stepping limit from 0.128s to 600s has a drawback because a different method of controlling the clock has to be used. It disables the kernel clock discipline and may have a negative impact on the clock accuracy. The `-x` option can be added to the `/etc/sysconfig/ntpd` configuration file.

15.4. Understanding the Drift File

The drift file is used to store the frequency offset between the system clock running at its nominal frequency and the frequency required to remain in synchronization with UTC. If present, the value contained in the drift file is read at system start and used to correct the clock source. Use of the drift file reduces the time required to achieve a stable and accurate time. The value is calculated, and the drift file replaced, once per hour by `ntpd`. The drift file is replaced, rather than just updated, and for this reason the drift file must be in a directory for which the `ntpd` has write permissions.

15.5. UTC, Timezones, and DST

As NTP is entirely in UTC (Universal Time, Coordinated), Timezones and DST (Daylight Saving Time) are applied locally by the system. The file `/etc/localtime` is a copy of, or symlink to, a zone information file from `/usr/share/zoneinfo`. The RTC may be in localtime or in UTC, as specified by the 3rd line of `/etc/adjtime`, which will be one of LOCAL or UTC to indicate how the RTC clock has been set. Users can easily change this setting using the checkbox **System Clock Uses UTC** in the **Date and Time** graphical configuration tool. See [Chapter 3, Configuring the Date and Time](#) for information on how to use that tool. Running the RTC in UTC is recommended to avoid various problems when daylight saving time is changed.

The operation of `ntpd` is explained in more detail in the man page `ntpd(8)`. The resources section lists useful sources of information. See [Section 15.20, “Additional Resources”](#).

15.6. 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](#)³. 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 15.17.12, “Configuring Symmetric Authentication Using a Key”](#) for an explanation of the **key** option.

15.7. 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 Fedora with **KVM** the default clock source is **kvm-clock**. See the [KVM guest timing management](#)⁴ chapter of the *Virtualization Host Configuration and Guest Installation Guide*.

15.8. 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 \(IERS\)](#)⁵. However, these announcements are important only to administrators of Stratum 1 servers because NTP transmits information about pending leap seconds and applies them automatically.

³ <http://www.rfc-editor.org/info/rfc5906>

⁴ http://docs.fedoraproject.org/en-US/Fedora/13/html/Virtualization_Guide/chap-Virtualization-KVM_guest_timing_management.html

⁵ <http://www.iers.org>

15.9. 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 15.17, “Configure NTP”](#), and more verbosely in the `ntp.conf(5)` man page.

Here follows a brief explanation of the contents of the default configuration file:

The driftfile entry

A path to the drift file is specified, the default entry on Fedora is:

```
driftfile /var/lib/ntp/drift
```

If you change this be certain that the directory is writable by `ntpd`. The file contains one value used to adjust the system clock frequency after every system or service start. See [Understanding the Drift File](#) for more information.

The access control entries

The following line sets the default access control restriction:

```
restrict default nomodify notrap nopeer noquery
```

- The **nomodify** options prevents any changes to the configuration.
- The **notrap** option prevents `ntpd` control message protocol traps.
- The **nopeer** option prevents a peer association being formed.
- The **noquery** option prevents `ntp` and `ntpd` queries, but not time queries, from being answered.



Important

The `ntp` and `ntpd` queries can be used in amplification attacks, therefore do not remove the **noquery** option from the **restrict default** command on publicly accessible systems.

See [CVE-2013-5211](#)⁶ for more details.

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.
```

⁶ <https://access.redhat.com/security/cve/CVE-2013-5211>


```
restrict 127.0.0.1
restrict ::1
```

Addresses can be added underneath if specifically required by another application.

Hosts on the local network are not permitted because of the "restrict default" line above. To change this, for example to allow hosts from the 192.0.2.0/24 network to query the time and statistics but nothing more, a line in the following format is required:

```
restrict 192.0.2.0 mask 255.255.255.0 nomodify notrap nopeer
```

To allow unrestricted access from a specific host, for example 192.0.2.250/32, 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, the **ntp.conf** file contains four public server entries:

```
server 0.fedora.pool.ntp.org iburst
server 1.fedora.pool.ntp.org iburst
server 2.fedora.pool.ntp.org iburst
server 3.fedora.pool.ntp.org iburst
```

The broadcast multicast servers entry

By default, the **ntp.conf** file contains some commented out examples. These are largely self explanatory. See [Section 15.17, "Configure NTP"](#) for the explanation of the specific commands. 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**.

15.10. Understanding the ntpd Sysconfig File

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

```
# Command line options for ntpd
OPTIONS="-g"
```

The **-g** option enables ntpd to ignore the offset limit of 1000s and attempt to synchronize the time even if the offset is larger than 1000s, but only on system start. Without that option **ntpd** will exit if the time offset is greater than 1000s. It will also exit after system start if the service is restarted and the offset is greater than 1000s even with the **-g** option.

15.11. Disabling chrony

In order to use `ntpd` the default user space daemon, `chronyd`, must be stopped and disabled. Issue the following command as root:

```
~]# systemctl stop chronyd
```

To prevent it restarting at system start, issue the following command as root:

```
~]# systemctl disable chronyd
```

To check the status of `chronyd`, issue the following command:

```
~]$ systemctl status chronyd
```

15.12. Checking if the NTP Daemon is Installed

To check if `ntpd` is installed, enter the following command as root:

```
~]# dnf install ntp
```

NTP is implemented by means of the daemon or service `ntpd`, which is contained within the `ntp` package.

15.13. Installing the NTP Daemon (ntpd)

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

```
~]# dnf install ntp
```

To enable `ntpd` at system start, enter the following command as root:

```
~]# systemctl enable ntpd
```

15.14. Checking the Status of NTP

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

```
~]$ systemctl status ntpd
```

To obtain a brief status report from `ntpd`, issue the following command:

```
~]$ ntpstat
unsynchronised
  time server re-starting
  polling server every 64 s
```

```
~]$ ntpstat
synchronised to NTP server (10.5.26.10) at stratum 2
  time correct to within 52 ms
  polling server every 1024 s
```


15.15. Configure the Firewall to Allow Incoming NTP Packets

The NTP traffic consists of UDP packets on port **123** and needs to be permitted through network and host-based firewalls in order for NTP to function.

Check if the firewall is configured to allow incoming NTP traffic for clients using the graphical **Firewall Configuration** tool.

To start the graphical **firewall-config** tool, press the **Super** key to enter the Activities Overview, type **firewall** and then press **Enter**. The **Firewall Configuration** window opens. You will be prompted for your user password.

To start the graphical firewall configuration tool using the command line, enter the following command as root user:

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

The **Firewall Configuration** window opens. Note, this command can be run as normal user but you will then be prompted for the root password from time to time.

Look for the word “Connected” in the lower left corner. This indicates that the **firewall-config** tool is connected to the user space daemon, **firewalld**.

15.15.1. Change the Firewall Settings

To immediately change the current firewall settings, ensure the drop-down selection menu labeled **Configuration** is set to **Runtime**. Alternatively, to edit the settings to be applied at the next system start, or firewall reload, select **Permanent** from the drop-down list.



Note

When making changes to the firewall settings in **Runtime** mode, your selection takes immediate effect when you set or clear the check box associated with the service. You should keep this in mind when working on a system that may be in use by other users.

When making changes to the firewall settings in **Permanent** mode, your selection will only take effect when you reload the firewall or the system restarts. To reload the firewall, select the **Options** menu and select **Reload Firewall**.

15.15.2. Open Ports in the Firewall for NTP Packets

To permit traffic through the firewall to a certain port, start the **firewall-config** tool and select the network zone whose settings you want to change. Select the **Ports** tab and then click the **Add** button. The **Port and Protocol** window opens.

Enter the port number **123** and select **udp** from the drop-down list.

15.16. Configure ntpdate Servers

The purpose of the **ntpdate** service is to set the clock during system boot. This was used previously to ensure that the services started after **ntpdate** would have the correct time and not observe a jump

in the clock. The use of `ntpd` and the list of step-tickers is considered deprecated and so Fedora uses the **-g** option to the **ntpd** command and not `ntpd` by default.

The `ntpd` service in Fedora is mostly useful only when used alone without `ntpd`. With **systemd**, which starts services in parallel, enabling the `ntpd` service will not ensure that other services started after it will have correct time unless they specify an ordering dependency on **time-sync.target**, which is provided by the `ntpd` service. The `ntp-wait` service (in the *ntp-perl* subpackage) provides the **time-sync** target for the `ntpd` service. In order to ensure a service starts with correct time, add **After=time-sync.target** to the service and enable one of the services which provide the target (`ntpd` or **sntp**, or **ntp-wait** if `ntpd` is enabled). Some services on Fedora have the dependency included by default (for example, `dhcpcd`, `dhcpcd6`, and `crond`).

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

```
~]$ systemctl status ntpdate
```

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

```
~]# systemctl enable ntpdate
```

In Fedora the default `/etc/ntp/step-tickers` file contains **0.fedora.pool.ntp.org**. To configure additional `ntpd` servers, using a text editor running as root, edit `/etc/ntp/step-tickers`. 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?

15.17. Configure NTP

To change the default configuration of the NTP service, use a text editor running as root user to edit the `/etc/ntp.conf` file. This file is installed together with `ntpd` and is configured to use time servers from the Fedora 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.

15.17.1. Configure Access Control to an NTP Service

To restrict or control access to the NTP service running on a system, make use of the **restrict** command in the `ntp.conf` file. See the commented out example:

```
# Hosts on local network are less restricted.
#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
```

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

```
restrict option
```

where *option* is one or more of:

- **ignore** — All packets will be ignored, including `ntp` 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 ntpdc queries are not affected. For more information on the **discard** command and the default values, see [Section 15.17.2, “Configure Rate Limiting Access to an NTP Service”](#).
- **lowpriotrap** — traps set by matching hosts to be low priority.
- **nomodify** — prevents any changes to the configuration.
- **noquery** — prevents ntpq and ntpdc queries, but not time queries, from being answered.
- **nopeer** — prevents a peer association being formed.
- **noserve** — deny all packets except ntpq and ntpdc queries.
- **notrap** — prevents ntpdc control message protocol traps.
- **notrust** — deny packets that are not cryptographically authenticated.
- **ntpport** — modify the match algorithm to only apply the restriction if the source port is the standard NTP UDP port **123**.
- **version** — deny packets that do not match the current NTP version.

To configure rate limit access to not respond at all to a query, the respective **restrict** command has to have the **limited** option. If ntpd should reply with a **KoD** packet, the **restrict** command needs to have both **limited** and **kod** options.

The ntpq and ntpdc queries can be used in amplification attacks (see [CVE-2013-5211](#)⁷ for more details), do not remove the **noquery** option from the **restrict default** command on publicly accessible systems.

15.17.2. Configure Rate Limiting Access to an NTP Service

To enable rate limiting access to the NTP service running on a system, add the **limited** option to the **restrict** command as explained in [Section 15.17.1, “Configure Access Control to an NTP Service”](#). If you do not want to use the default discard parameters, then also use the **discard** command as explained here.

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

```
discard [average value] [minimum value] [monitor value]
```

- **average** — specifies the minimum average packet spacing to be permitted, it accepts an argument in \log_2 seconds. The default value is 3 (2^3 equates to 8 seconds).
- **minimum** — specifies the minimum packet spacing to be permitted, it accepts an argument in \log_2 seconds. The default value is 1 (2^1 equates to 2 seconds).
- **monitor** — specifies the discard probability for packets once the permitted rate limits have been exceeded. The default value is 3000 seconds. This option is intended for servers that receive 1000 or more requests per second.

⁷ <https://access.redhat.com/security/cve/CVE-2013-5211>

Examples of the **discard** command are as follows:

```
discard average 4
```

```
discard average 4 minimum 2
```

15.17.3. Adding a Peer Address

To add the address of a peer, that is to say, the address of a server running an NTP service of the same stratum, make use of the **peer** command in the **ntp.conf** file.

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

```
peer address
```

where *address* is an IP unicast address or a DNS resolvable name. The address must only be that of a system known to be a member of the same stratum. Peers should have at least one time source that is different to each other. Peers are normally systems under the same administrative control.

15.17.4. Adding a Server Address

To add the address of a server, that is to say, the address of a server running an NTP service of a higher stratum, make use of the **server** command in the **ntp.conf** file.

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

```
server address
```

where *address* is an IP unicast address or a DNS resolvable name. The address of a remote reference server or local reference clock from which packets are to be received.

15.17.5. Adding a Broadcast or Multicast Server Address

To add a broadcast or multicast address for sending, that is to say, the address to broadcast or multicast NTP packets to, make use of the **broadcast** command in the **ntp.conf** file.

The broadcast and multicast modes require authentication by default. See [Section 15.6, “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](#)⁸.

⁸ <http://www.rfc-editor.org/info/rfc2365>

15.17.6. Adding a Manycast Client Address

To add a manycast client address, that is to say, to configure a multicast address to be used for NTP server discovery, make use of the **manycastclient** command in the **ntp.conf** file.

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

```
manycastclient address
```

where *address* is an IP multicast address from which packets are to be received. The client will send a request to the address and select the best servers from the responses and ignore other servers. NTP communication then uses unicast associations, as if the discovered NTP servers were listed in **ntp.conf**.

This command configures a system to act as an NTP client. Systems can be both client and server at the same time.

15.17.7. Adding a Broadcast Client Address

To add a broadcast client address, that is to say, to configure a broadcast address to be monitored for broadcast NTP packets, make use of the **broadcastclient** command in the **ntp.conf** file.

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

```
broadcastclient
```

Enables the receiving of broadcast messages. Requires authentication by default. See [Section 15.6, “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.

15.17.8. Adding a Manycast Server Address

To add a manycast server address, that is to say, to configure an address to allow the clients to discover the server by multicasting NTP packets, make use of the **manycastserver** command in the **ntp.conf** file.

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

```
manycastserver address
```

Enables the sending of multicast messages. Where *address* is the address to multicast to. This should be used together with authentication to prevent service disruption.

This command configures a system to act as an NTP server. Systems can be both client and server at the same time.

15.17.9. Adding a Multicast Client Address

To add a multicast client address, that is to say, to configure a multicast address to be monitored for multicast NTP packets, make use of the **multicastclient** command in the **ntp.conf** file.

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

```
multicastclient address
```


Enables the receiving of multicast messages. Where *address* is the address to subscribe to. This should be used together with authentication to prevent service disruption.

This command configures a system to act as an NTP client. Systems can be both client and server at the same time.

15.17.10. Configuring the Burst Option

Using the **burst** option against a public server is considered abuse. Do not use this option with public NTP servers. Use it only for applications within your own organization.

To increase the average quality of time offset statistics, add the following option to the end of a server command:

```
burst
```

At every poll interval, 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.

15.17.11. Configuring the iburst Option

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

```
iburst
```

At every poll interval, send a burst of eight packets instead of one. When the server is not responding, packets are sent 16s apart. When the server responds, packets are sent every 2s. For use with the **server** command to reduce the time taken for initial synchronization. This is now a default option in the configuration file.

15.17.12. Configuring Symmetric Authentication Using a Key

To configure symmetric authentication using a key, add the following option to the end of a server or peer command:

```
key number
```

where *number* is in the range **1** to **65534** inclusive. This option enables the use of a *message authentication code* (MAC) in packets. This option is for use with the **peer**, **server**, **broadcast**, and **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 15.6, “Authentication Options for NTP”](#).

15.17.13. Configuring the Poll Interval

To change the default poll interval, add the following options to the end of a server or peer command:


```
minpoll value and maxpoll value
```

Options to change the default poll interval, where the interval in seconds will be calculated by raising 2 to the power of *value*, in other words, the interval is expressed in \log_2 seconds. The default **minpoll** value is 6, 2^6 equates to 64s. The default value for **maxpoll** is 10, which equates to 1024s. Allowed values are in the range 3 to 17 inclusive, which equates to 8s to 36.4h respectively. These options are for use with the **peer** or **server**. Setting a shorter **maxpoll** may improve clock accuracy.

15.17.14. Configuring Server Preference

To specify that a particular server should be preferred above others of similar statistical quality, add the following option to the end of a server or peer command:

```
prefer
```

Use this server for synchronization in preference to other servers of similar statistical quality. This option is for use with the **peer** or **server** commands.

15.17.15. Configuring the Time-to-Live for NTP Packets

To specify that a particular *time-to-live* (TTL) value should be used in place of the default, add the following option to the end of a server or peer command:

```
ttl value
```

Specify the time-to-live value to be used in packets sent by broadcast servers and multicast NTP servers. Specify the maximum time-to-live value to use for the “expanding ring search” by a manycast client. The default value is **127**.

15.17.16. Configuring the NTP Version to Use

To specify that a particular version of NTP should be used in place of the default, add the following option to the end of a server or peer command:

```
version value
```

Specify the version of NTP set in created NTP packets. The value can be in the range **1** to **4**. The default is **4**.

15.18. Configuring the Hardware Clock Update

To configure the system clock to update the hardware clock, also known as the real-time clock (RTC), once after executing **ntpdate**, add the following line to **/etc/sysconfig/ntpdate**:

```
SYNC_HWCLOCK=yes
```

To update the hardware clock from the system clock, issue the following command as root:

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

When the system clock is being synchronized by **ntpd** or **chronyd**, the kernel will in turn update the RTC every 11 minutes automatically.

15.19. Configuring Clock Sources

To list the available clock sources on your system, issue the following commands:

```
~]$ cd /sys/devices/system/clocksource/clocksource0/
clocksource0]$ cat available_clocksource
kvm-clock tsc hpet acpi_pm
clocksource0]$ cat current_clocksource
kvm-clock
```

In the above example, the kernel is using **kvm-clock**. This was selected at boot time as this is a virtual machine.

To override the default clock source, append the **clocksource** directive to the GRUB_CMDLINE_LINUX line in the **/etc/default/grub** file and rebuild the **grub.cfg** file. For example:

```
GRUB_CMDLINE_LINUX="rd.lvm.lv=rhel/root crashkernel=auto rd.lvm.lv=rhel/swap
vconsole.font=latarcyrheb-sun16 vconsole.keymap=us rhgb quiet clocksource=tsc"
```

The available clock source is architecture dependent.

Rebuild the **grub.cfg** file as follows:

- On BIOS-based machines, issue the following command as root:

```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

- On UEFI-based machines, issue the following command as root:

```
~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

15.20. Additional Resources

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

15.20.1. Installed Documentation

- **ntpd(8)** man page — Describes ntpd in detail, including the command line options.
- **ntp.conf(5)** man page — Contains information on how to configure associations with servers and peers.
- **ntpq(8)** man page — Describes the NTP query utility for monitoring and querying an NTP server.
- **ntpdcc(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.
- **ntp_decode(5)** man page — Lists the status words, event messages and error codes used for ntpd reporting and monitoring.
- **ntpstat(8)** man page — Describes a utility for reporting the synchronization state of the NTP daemon running on the local machine.
- **ntptime(8)** man page — Describes a utility for reading and setting kernel time variables.
- **tickadj(8)** man page — Describes a utility for reading, and optionally setting, the length of the tick.

15.20.2. Useful Websites

<http://doc.ntp.org/>

The NTP Documentation Archive

<http://www.eecis.udel.edu/~mills/ntp.html>

Network Time Synchronization Research Project.

<http://www.eecis.udel.edu/~mills/ntp/html/manyopt.html>

Information on Automatic Server Discovery in NTPv4.

Configuring PTP Using ptp4l

16.1. 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 Fedora 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).

16.1.1. 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.

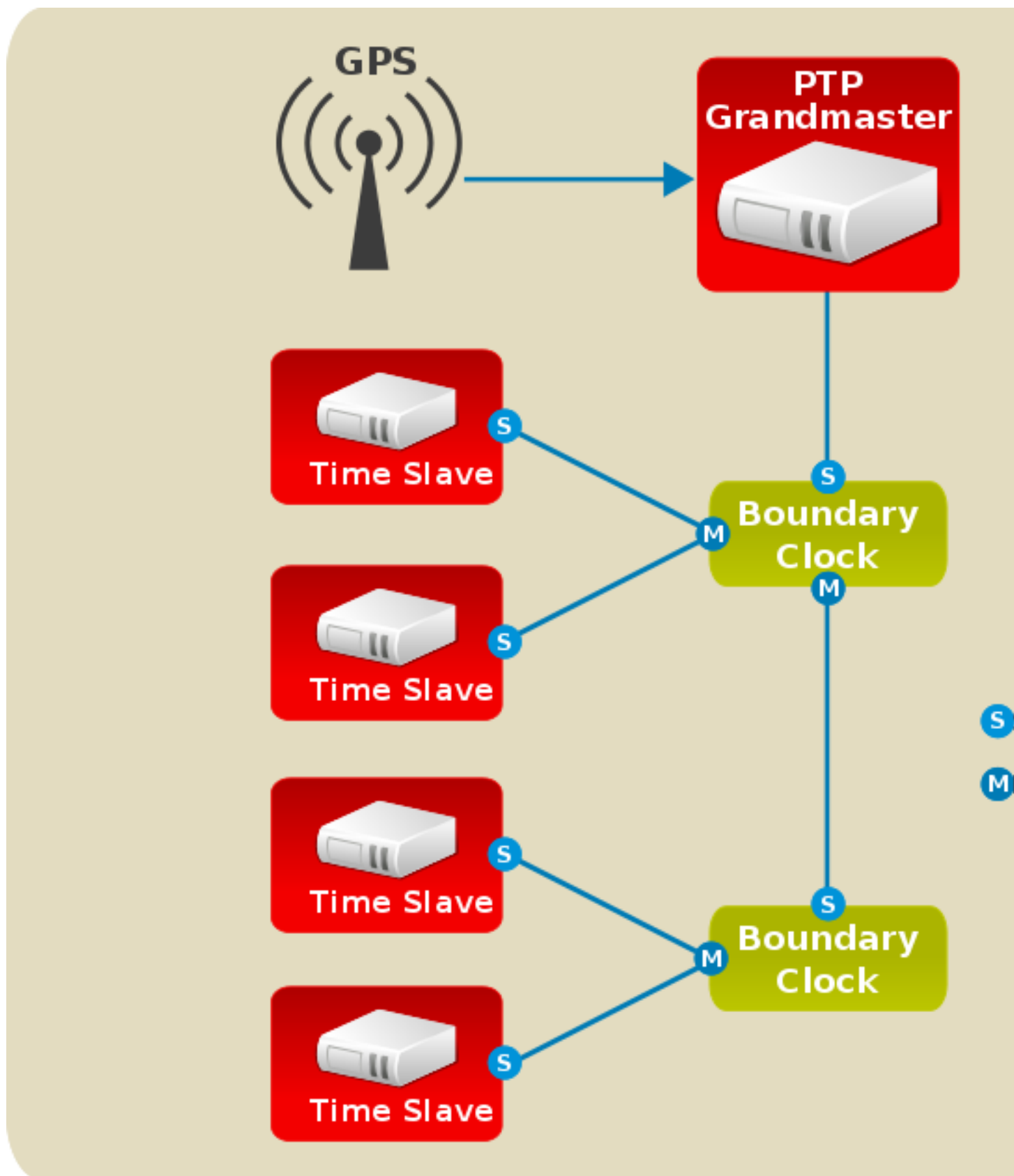


Figure 16.1. PTP grandmaster, boundary, and slave Clocks

16.1.2. 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.

16.2. 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.

16.2.1. 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 em3
Time stamping parameters for em3:
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 *em3* 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`

16.2.2. Installing PTP

The kernel in Fedora 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:

```
~]# dnf 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 16.7, “Serving PTP Time with NTP”](#).

16.2.3. Starting ptp4l

The **ptp4l** program can be started from the command line or it can be started as a service. When running as a service, options are specified in the `/etc/sysconfig/ptp4l` file. Options required for use both by the service and on the command line should be specified in the `/etc/ptp4l.conf` file. The `/etc/sysconfig/ptp4l` file includes the `-f /etc/ptp4l.conf` command line option, which causes the **ptp4l** program to read the `/etc/ptp4l.conf` file and process the options it contains. The use of the `/etc/ptp4l.conf` is explained in [Section 16.3, “Specifying a Configuration File”](#). More information on the different **ptp4l** options and the configuration file settings can be found in the **ptp4l(8)** man page.

Starting ptp4l as a Service

To start **ptp4l** as a service, issue the following command as root:

```
~]# systemctl start ptp4l
```

Using ptp4l From The Command Line

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 em3 -m
```

Where `em3` 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 em3 -m
selected em3 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 **adj** value is the frequency adjustment of the clock in parts per billion (ppb). The path delay value is the estimated delay of the synchronization messages sent from the master in nanoseconds. Port 0 is a Unix domain socket used for local PTP management. Port 1 is the em3 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:

```
~]# systemctl start ptp4l
```

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 em3 -m -S
```

16.2.3.1. 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. Warnings will be printed in the following circumstances:

- When a peer delay request is received on a port using the E2E mechanism.
- When a E2E delay request is received on a port using the P2P mechanism.

16.3. 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 em3 -m -S** options shown above would look as follows:

```
~]# cat /etc/ptp4l.conf
[global]
verbose          1
time_stamping    software
[em3]
```

16.4. 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.ffff.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.ffff.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.ffff.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.

16.5. 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:

```
~]# systemctl restart phc2sys
```

Under normal circumstances, use **systemctl** 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 em3 -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 35 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 em3 -0 -35
```

Once the **phc2sys** servo is in a locked state, the clock will not be stepped, unless the **-S** option is used. This means that the **phc2sys** program should be started after the **ptp4l** program has synchronized the PTP hardware clock. However, with **-w**, it is not necessary to start **phc2sys** after **ptp4l** as it will wait for it to synchronize the clock.

The **phc2sys** program can also be started as a service by running:

```
~]# systemctl start phc2sys
```

When running as a service, options are specified in the **/etc/sysconfig/phc2sys** file. More information on the different **phc2sys** options can be found in the **phc2sys(8)** man page.

Note that the examples in this section assume the command is run on a slave system or slave port.

16.6. Verifying Time Synchronization

When PTP time synchronization is working properly, new messages with offsets and frequency adjustments will be printed periodically to the **ptp4l** and **phc2sys** (if hardware time stamping is used) outputs. These values will eventually converge after a short period of time. These messages can be seen in **/var/log/messages** file. An example of the output follows:

```
ptp4l[352.359]: selected /dev/ptp0 as PTP clock
ptp4l[352.361]: port 1: INITIALIZING to LISTENING on INITIALIZE
ptp4l[352.361]: port 0: INITIALIZING to LISTENING on INITIALIZE
ptp4l[353.210]: port 1: new foreign master 00a069.ffff.0b552d-1
ptp4l[357.214]: selected best master clock 00a069.ffff.0b552d
ptp4l[357.214]: port 1: LISTENING to UNCALIBRATED on RS_SLAVE
ptp4l[359.224]: master offset      3304 s0 freq      +0 path delay      9202
ptp4l[360.224]: master offset      3708 s1 freq     -29492 path delay      9202
ptp4l[361.224]: master offset     -3145 s2 freq     -32637 path delay      9202
ptp4l[361.224]: port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l[362.223]: master offset      -145 s2 freq     -30580 path delay      9202
ptp4l[363.223]: master offset      1043 s2 freq     -29436 path delay      8972
ptp4l[364.223]: master offset       266 s2 freq     -29900 path delay      9153
ptp4l[365.223]: master offset       430 s2 freq     -29656 path delay      9153
ptp4l[366.223]: master offset       615 s2 freq     -29342 path delay      9169
ptp4l[367.222]: master offset      -191 s2 freq     -29964 path delay      9169
ptp4l[368.223]: master offset       466 s2 freq     -29364 path delay      9170
ptp4l[369.235]: master offset        24 s2 freq     -29666 path delay      9196
ptp4l[370.235]: master offset     -375 s2 freq     -30058 path delay      9238
ptp4l[371.235]: master offset       285 s2 freq     -29511 path delay      9199
ptp4l[372.235]: master offset      -78 s2 freq     -29788 path delay      9204
```

An example of the **phc2sys** output follows:

```
phc2sys[526.527]: Waiting for ptp4l...
phc2sys[527.528]: Waiting for ptp4l...
phc2sys[528.528]: phc offset      55341 s0 freq      +0 delay      2729
```



```

phc2sys[529.528]: phc offset      54658 s1 freq -37690 delay 2725
phc2sys[530.528]: phc offset       888 s2 freq -36802 delay 2756
phc2sys[531.528]: phc offset     1156 s2 freq -36268 delay 2766
phc2sys[532.528]: phc offset      411 s2 freq -36666 delay 2738
phc2sys[533.528]: phc offset      -73 s2 freq -37026 delay 2764
phc2sys[534.528]: phc offset       39 s2 freq -36936 delay 2746
phc2sys[535.529]: phc offset       95 s2 freq -36869 delay 2733
phc2sys[536.529]: phc offset     -359 s2 freq -37294 delay 2738
phc2sys[537.529]: phc offset     -257 s2 freq -37300 delay 2753
phc2sys[538.529]: phc offset      119 s2 freq -37001 delay 2745
phc2sys[539.529]: phc offset      288 s2 freq -36796 delay 2766
phc2sys[540.529]: phc offset     -149 s2 freq -37147 delay 2760
phc2sys[541.529]: phc offset     -352 s2 freq -37395 delay 2771
phc2sys[542.529]: phc offset      166 s2 freq -36982 delay 2748
phc2sys[543.529]: phc offset       50 s2 freq -37048 delay 2756
phc2sys[544.530]: phc offset      -31 s2 freq -37114 delay 2748
phc2sys[545.530]: phc offset     -333 s2 freq -37426 delay 2747
phc2sys[546.530]: phc offset      194 s2 freq -36999 delay 2749

```

For **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.fffe.0b552d-1
ptp4l: [619.574] selected best master clock 00a069.fffe.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 em3 -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

```


16.7. 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`.

16.8. 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
[em3]
# ptp4l -f /etc/ptp4l.conf
```

With hardware time stamping, **phc2sys** needs to be used to synchronize the PTP hardware clock to the system clock. If running **phc2sys** as a service, edit the `/etc/sysconfig/phc2sys` configuration file. The default setting in the `/etc/sysconfig/phc2sys` file is as follows:

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

As root, edit that line as follows:

```
~]# vi /etc/sysconfig/phc2sys
OPTIONS="-a -r -r"
```

The **-r** option is used twice here to allow synchronization of the PTP hardware clock on the NIC from the system clock. Restart the **phc2sys** service for the changes to take effect:

```
~]# systemctl restart phc2sys
```

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 for the PI servo:

```
~]# phc2sys -a -r -r -P 0.01 -I 0.0001
```


16.9. 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/**.

16.9.1. Starting timemaster as a Service

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

```
~]# systemctl start timemaster
```

This will read the options in **/etc/timemaster.conf**.

16.9.2. Understanding the timemaster Configuration File

Fedora 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 15, 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**.
- **[ptp4l]** — 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.

16.9.3. Configuring timemaster Options

Procedure 16.1. 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 15, 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 [Chapter 14, 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 [Chapter 14, 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 15, 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 **phc2sys(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:

```
~]# systemctl restart timemaster
```

16.10. 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.fc21** 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:

```
~]# systemctl restart ptp4l
```

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:

```
~]# systemctl restart phc2sys
```

16.11. Additional Resources

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

16.11.1. 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).

- **timemaster(8)** man page — Describes a program that uses **ptp4l** and **phc2sys** to synchronize the system clock using chronyd or ntpd.

16.11.2. Useful Websites

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

The Linux PTP project.

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

The IEEE 1588 Standard.

Part V. Monitoring and Automation

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

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.

17.1. Viewing System Processes

17.1.1. 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        0:02 /usr/lib/systemd/systemd --system --deserialize 20
    2 ?           S          0:00 [kthreadd]
    3 ?           S          0:00 [ksoftirqd/0]
    5 ?           S          0:00 [kworker/u:0]
    6 ?           S          0:00 [migration/0]
[output truncated]
```

To display the owner alongside each process, use the following command:

```
ps aux
```

Apart from the information provided by the **ps ax** command, **ps aux** displays the effective username of the process owner (**USER**), the percentage of the CPU (**%CPU**) and memory (**%MEM**) usage, the virtual memory size in kilobytes (**VSZ**), the non-swapped physical memory size in kilobytes (**RSS**), and the time or date the process was started. For instance:

```
~]$ ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.3  53128  2988 ?        Ss   13:28   0:02 /usr/lib/systemd/systemd --system --deserialize 20
root         2  0.0  0.0      0     0 ?        S    13:28   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    13:28   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S    13:28   0:00 [kworker/u:0]
root         6  0.0  0.0      0     0 ?        S    13:28   0:00 [migration/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
2625 ?        S1          0:00 emacs
```

For a complete list of available command line options, refer to the **ps(1)** manual page.

17.1.2. Using the top Command

The **top** command displays a real-time list of processes that are running on the system. It also displays additional information about the system uptime, current CPU and memory usage, or total number of running processes, and allows you to perform actions such as sorting the list or killing a process.

To run the **top** command, type the following at a shell prompt:

```
top
```

For each listed process, the **top** command displays the process ID (**PID**), the effective username of the process owner (**USER**), the priority (**PR**), the nice value (**NI**), the amount of virtual memory the process uses (**VRT**), the amount of non-swapped physical memory the process uses (**RES**), the amount of shared memory the process uses (**SHR**), the percentage of the CPU (**%CPU**) and memory (**%MEM**) usage, the cumulated CPU time (**TIME+**), and the name of the executable file (**COMMAND**). For example:

```
~]$ top
top - 19:22:08 up 5:53, 3 users, load average: 1.08, 1.03, 0.82
Tasks: 117 total, 2 running, 115 sleeping, 0 stopped, 0 zombie
Cpu(s): 9.3%us, 1.3%sy, 0.0%ni, 85.1%id, 0.0%wa, 1.7%hi, 0.0%si, 2.6%st
Mem: 761956k total, 617256k used, 144700k free, 24356k buffers
Swap: 1540092k total, 55780k used, 1484312k free, 256408k cached

  PID USER      PR  NI  VIRT  RES  SHR  S %CPU  %MEM    TIME+  COMMAND
  510 john       20   0 1435m  99m  18m  S  9.0  13.3   3:30.52 gnome-shell
32686 root        20   0  156m  27m 3628  R  2.0   3.7   0:48.69 Xorg
 2625 john       20   0  488m  27m  14m  S  0.3   3.7   0:00.70 emacs
    1 root        20   0 53128 2640 1152  S  0.0   0.3   0:02.83 systemd
    2 root        20   0     0     0     0  S  0.0   0.0   0:00.01 kthreadd
    3 root        20   0     0     0     0  S  0.0   0.0   0:00.18 ksoftirqd/0
    5 root        20   0     0     0     0  S  0.0   0.0   0:00.00 kworker/u:0
    6 root        RT   0     0     0     0  S  0.0   0.0   0:00.00 migration/0
    7 root        RT   0     0     0     0  S  0.0   0.0   0:00.30 watchdog/0
    8 root         0 -20     0     0     0  S  0.0   0.0   0:00.00 cpuset
    9 root         0 -20     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 kdevtmpfs
   11 root         0 -20     0     0     0  S  0.0   0.0   0:00.00 netns
   12 root        20   0     0     0     0  S  0.0   0.0   0:00.11 sync_supers
   13 root        20   0     0     0     0  S  0.0   0.0   0:00.00 bdi-default
   14 root         0 -20     0     0     0  S  0.0   0.0   0:00.00 kintegrityd
   15 root         0 -20     0     0     0  S  0.0   0.0   0:00.00 kblockd
```

Table 17.1, “Interactive top commands” contains useful interactive commands that you can use with **top**. For more information, refer to the **top(1)** manual page.

Table 17.1. Interactive top commands

Command	Description
Enter, Space	Immediately refreshes the display.
h, ?	Displays a help screen.

Command	Description
k	Kills a process. You are prompted for the process ID and the signal to send to it.
n	Changes the number of displayed processes. You are prompted to enter the number.
u	Sorts the list by user.
M	Sorts the list by memory usage.
P	Sorts the list by CPU usage.
q	Terminates the utility and returns to the shell prompt.

17.1.3. Using the System Monitor Tool

The **Processes** tab of the **System Monitor** tool allows you to view, search for, change the priority of, and kill processes from the graphical user interface.

To start the **System Monitor** tool, either select **Applications** → **System Tools** → **System Monitor** from the **Activities** menu, or type **gnome-system-monitor** at a shell prompt. Then click the **Processes** tab to view the list of running processes.

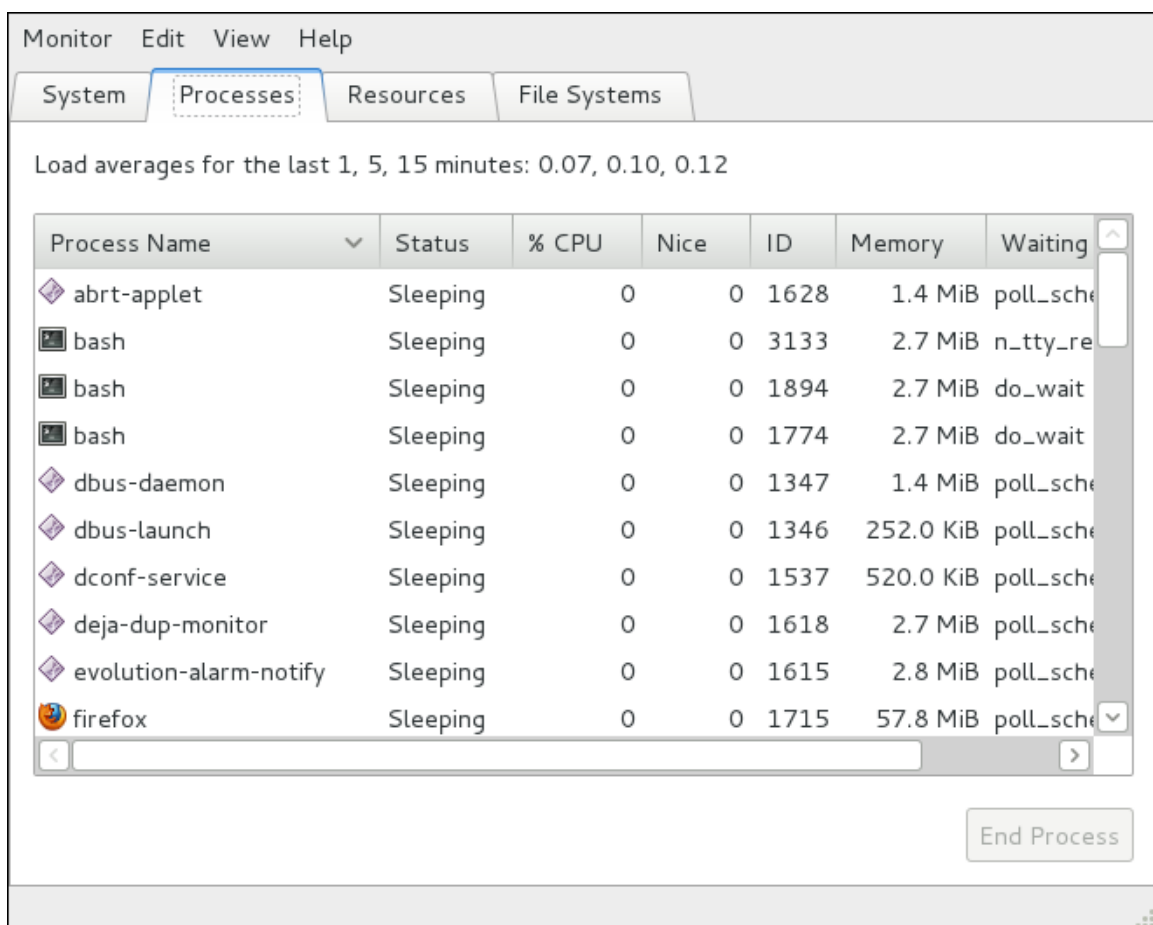


Figure 17.1. System Monitor — Processes

For each listed process, the **System Monitor** tool displays its name (**Process Name**), current status (**Status**), percentage of the memory 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.

17.2. Viewing Memory Usage

17.2.1. 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:	761956	607500	154456	0	37404	156176
-/+ buffers/cache:		413920	348036			
Swap:	1540092	84408	1455684			

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:	744	593	150	0	36	152
-/+ buffers/cache:		404	339			
Swap:	1503	82	1421			

For a complete list of available command line options, refer to the **free(1)** manual page.

17.2.2. 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 **Activities** menu, or type **gnome-system-monitor** at a shell prompt. Then click the **Resources** tab to view the system's memory usage.



Figure 17.2. 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.

17.3. Viewing CPU Usage

17.3.1. 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 **Activities** menu, or type **gnome-system-monitor** at a shell prompt. Then click the **Resources** tab to view the system's CPU usage.

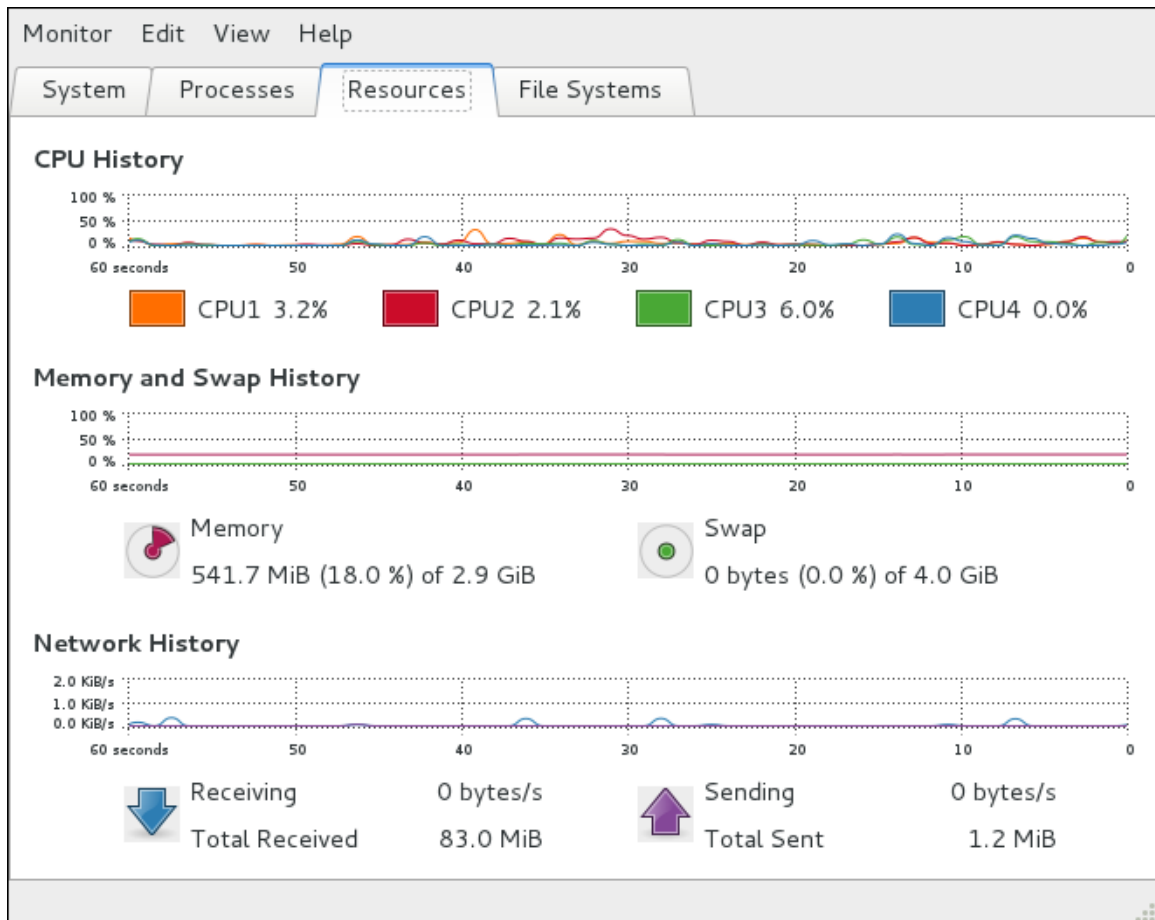


Figure 17.3. System Monitor — Resources

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.

17.4. Viewing Block Devices and File Systems

17.4.1. 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 (**MOUNTPPOINT**). For example:

```
~]$ lsblk
NAME                                MAJ:MIN RM  SIZE RO TYPE MOUNTPPOINT
sr0                                  11:0    1 1024M  0 rom
vda                                  252:0    0   20G  0 disk
|-vda1                              252:1    0   500M  0 part /boot
`-vda2                              252:2    0 19.5G  0 part
```



```
| -vg_fedora-lv_swap (dm-0) 253:0    0    1.5G  0 lvm  [SWAP]
|-vg_fedora-lv_root  (dm-1) 253:1    0    18G  0 lvm  /
```

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 disk
vda1                              252:1    0   500M  0 part /boot
vda2                              252:2    0   19.5G  0 part
vg_fedora-lv_swap (dm-0) 253:0    0   1.5G  0 lvm  [SWAP]
vg_fedora-lv_root  (dm-1) 253:1    0   18G  0 lvm  /
```

For a complete list of available command line options, refer to the **lsblk(8)** manual page.

17.4.2. 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="4ea24c68-ab10-47d4-8a6b-b8d3a002acba" TYPE="ext4"
/dev/vda2: UUID="iJ9YwJ-leFf-A1zb-VVaK-H9t1-raLW-HoqlUG" TYPE="LVM2_member"
/dev/mapper/vg_fedora-lv_swap: UUID="d6d755bc-3e3e-4e8f-9bb5-a5e7f4d86ffd" TYPE="swap"
/dev/mapper/vg_fedora-lv_root: LABEL="_Fedora-17-x86_6" UUID="77ba9149-751a-48e0-974f-ad94911734b9" TYPE="ext4"
```

By default, the **lsblk** command lists all available block devices. To display information about a particular device only, specify the device name on the command line:

```
blkid device_name
```

For instance, to display information about **/dev/vda1**, type:

```
~]# blkid /dev/vda1
/dev/vda1: UUID="4ea24c68-ab10-47d4-8a6b-b8d3a002acba" 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=4ea24c68-ab10-47d4-8a6b-b8d3a002acba
ID_FS_UUID_ENC=4ea24c68-ab10-47d4-8a6b-b8d3a002acba
ID_FS_VERSION=1.0
ID_FS_TYPE=ext4
ID_FS_USAGE=filesystem
ID_PART_ENTRY_SCHEME=dos
ID_PART_ENTRY_TYPE=0x83
ID_PART_ENTRY_FLAGS=0x80
ID_PART_ENTRY_NUMBER=1
ID_PART_ENTRY_OFFSET=2048
ID_PART_ENTRY_SIZE=1024000
ID_PART_ENTRY_DISK=252:0
```

For a complete list of available command line options, refer to the **blkid(8)** manual page.

17.4.3. Using the partx Command

The **partx** command allows you to display a list of disk partitions. To list the partition table of a particular disk, as root, run this command with the **-s** option followed by the device name:

```
partx -s device_name
```

For example, to list partitions on **/dev/vda**, type:

```
~]# partx -s /dev/vda
NR      START      END  SECTORS  SIZE NAME  UUID
 1      2048     1026047  1024000  500M
 2  1026048  41943039  40916992  19.5G
```

For a complete list of available command line options, refer to the **partx(8)** manual page.

17.4.4. 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_fedora-lv_root
|-/proc      proc        ext4     rw,relatime,seclabel,data=0
| `-/proc/sys/fs/binfmt_misc systemd-1  autofs   rw,relatime,fd=23,pgrp=1,ti
|-/sys       sysfs       sysfs    rw,nosuid,nodev,noexec,rela
|  |-/sys/kernel/security securityfs security rw,nosuid,nodev,noexec,rela
|  |-/sys/fs/selinux  selinuxfs  selinuxf rw,relatime
|  |-/sys/fs/cgroup   tmpfs      tmpfs    rw,nosuid,nodev,noexec,secl
```



```
| | |-/sys/fs/cgroup/systemd      cgroup    cgroup    rw,nosuid,nodev,noexec,rela
| | |-/sys/fs/cgroup/cpuset      cgroup    cgroup    rw,nosuid,nodev,noexec,rela
| | |-/sys/fs/cgroup/cpu,cpuacct cgroup    cgroup    rw,nosuid,nodev,noexec,rela
| | |-/sys/fs/cgroup/memory      cgroup    cgroup    rw,nosuid,nodev,noexec,rela
| | |-/sys/fs/cgroup/devices     cgroup    cgroup    rw,nosuid,nodev,noexec,rela
| | |-/sys/fs/cgroup/freezer     cgroup    cgroup    rw,nosuid,nodev,noexec,rela
| | |-/sys/fs/cgroup/net_cls     cgroup    cgroup    rw,nosuid,nodev,noexec,rela
| | |-/sys/fs/cgroup/blkio       cgroup    cgroup    rw,nosuid,nodev,noexec,rela
| | |-/sys/fs/cgroup/perf_event  cgroup    cgroup    rw,nosuid,nodev,noexec,rela
| | |-/sys/kernel/debug          debugfs    debugfs    rw,relatime
| | |-/sys/kernel/config         configfs   configfs   rw,relatime
[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,nosuid,nodev,noexec,relatime
/sys            sysfs       sysfs     rw,nosuid,nodev,noexec,relatime,s
/dev            devtmpfs    devtmpfs  rw,nosuid,seclabel,size=370080k,n
/dev/pts        devpts      devpts    rw,nosuid,noexec,relatime,seclabe
/dev/shm        tmpfs       tmpfs     rw,nosuid,nodev,seclabel
/run            tmpfs       tmpfs     rw,nosuid,nodev,seclabel,mode=755
/               /dev/mapper/vg_fedora-lv_root
                ext4        rw,relatime,seclabel,data=ordered
/sys/kernel/security securityfs   security  rw,nosuid,nodev,noexec,relatime
/sys/fs/selinux selinuxfs   selinuxf  rw,relatime
/sys/fs/cgroup  tmpfs       tmpfs     rw,nosuid,nodev,noexec,seclabel,m
/sys/fs/cgroup/systemd cgroup      cgroup    rw,nosuid,nodev,noexec,relatime,r
[output truncated]
```

You can also choose to list only file systems of a particular type. To do so, add the **-t** command line option followed by a file system type:

```
findmnt -t type
```

For example, to all list ext4 file systems, type:

```
~]$ findmnt -t ext4
TARGET SOURCE          FSTYPE  OPTIONS
/       /dev/mapper/vg_fedora-lv_root ext4     rw,relatime,seclabel,data=ordered
/boot   /dev/vda1          ext4     rw,relatime,seclabel,data=ordered
```

For a complete list of available command line options, refer to the **findmnt(8)** manual page.

17.4.5. 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
rootfs                18877356 4605476  14082844 25% /
devtmpfs              370080    0      370080  0% /dev
tmpfs                 380976    256    380720  1% /dev/shm
tmpfs                 380976    3048    377928  1% /run
/dev/mapper/vg_fedora-lv_root 18877356 4605476  14082844 25% /
tmpfs                 380976    0      380976  0% /sys/fs/cgroup
tmpfs                 380976    0      380976  0% /media
/dev/vda1             508745    85018    398127 18% /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
rootfs          19G   4.4G   14G   25% /
devtmpfs        362M     0   362M    0% /dev
tmpfs           373M   256K   372M    1% /dev/shm
tmpfs           373M   3.0M   370M    1% /run
/dev/mapper/vg_fedora-lv_root 19G   4.4G   14G   25% /
tmpfs           373M     0   373M    0% /sys/fs/cgroup
tmpfs           373M     0   373M    0% /media
/dev/vda1       497M    84M   389M   18% /boot
```

Note that the **/dev/shm** entry represents the system's virtual memory file system, **/sys/fs/cgroup** is a cgroup file system, and **/run** contains information about the running system.

For a complete list of available command line options, refer to the **df(1)** manual page.

17.4.6. 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
8  ./gconf/apps/gnome-terminal/profiles/Default
12 ./gconf/apps/gnome-terminal/profiles
16 ./gconf/apps/gnome-terminal
[output truncated]
```



```
460    ./gimp-2.6
68828  .
```

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
8.0K    ./gconf/apps/gnome-terminal/profiles/Default
12K     ./gconf/apps/gnome-terminal/profiles
16K     ./gconf/apps/gnome-terminal
[output truncated]
460K    ./gimp-2.6
68M     .
```

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
68M     .
```

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

17.4.7. 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 **Activities** menu, or type **gnome-system-monitor** at a shell prompt. Then click the **File Systems** tab to view a list of file systems.

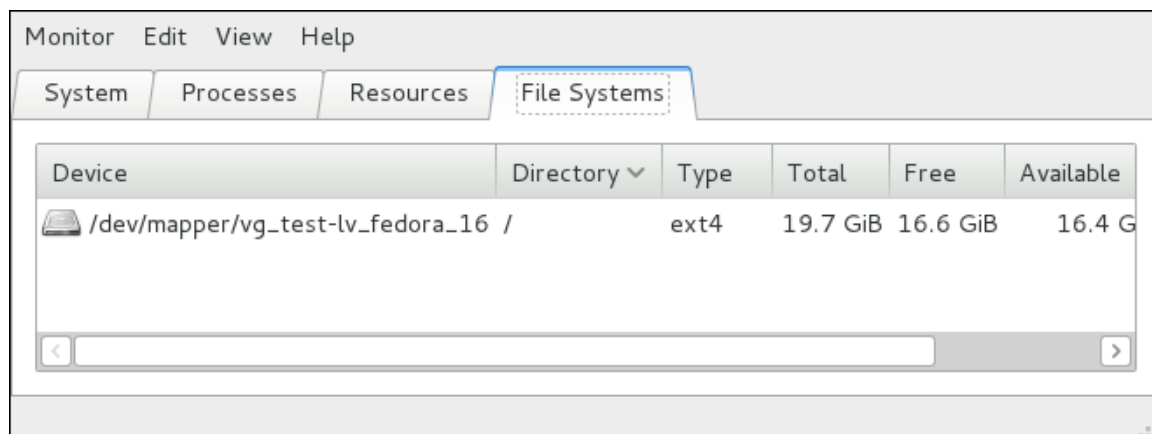


Figure 17.4. System Monitor — File Systems

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**).

17.5. Viewing Hardware Information

17.5.1. Using the `lspci` Command

The **lspci** command lists all PCI devices that are present in the system:

```
lspci
```

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, refer to the **lspci(8)** manual page.

17.5.2. 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, refer to the **lsusb(8)** manual page.

17.5.3. 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, refer to the `pccardctl(8)` manual page.

17.5.4. 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, refer to the `lscpu(1)` manual page.

17.6. Monitoring Performance with Net-SNMP

Fedora 26 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.

17.6.1. Installing Net-SNMP

The Net-SNMP software suite is available as a set of RPM packages in the Fedora software distribution. [Table 17.2, “Available Net-SNMP packages”](#) summarizes each of the packages and their contents.

Table 17.2. Available Net-SNMP packages

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

To install any of these packages, use the **dnf** command in the following form:

```
dnf 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:

```
~]# dnf 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 Fedora, refer to [Section 6.2.4, “Installing Packages”](#).

17.6.2. 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 or disable it in the **multi-user** target unit. For more information on the concept of target units and how to manage system services in Fedora in general, refer to [Chapter 7, Services and Daemons](#).

17.6.2.1. Starting the Service

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

```
systemctl start snmpd.service
```

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

```
systemctl enable snmpd.service
```

This will enable the service in the **multi-user** target unit.

17.6.2.2. Stopping the Service

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

```
systemctl stop snmpd.service
```

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

```
systemctl disable snmpd.service
```

This will disable the service in the `multi-user` target unit.

17.6.2.3. Restarting the Service

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

```
systemctl restart snmpd.service
```

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:

```
systemctl reload snmpd.service
```

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

17.6.3. 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 Fedora 26 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, refer to the `snmpd.conf(5)` manual page. Additionally, there is a utility in the `net-snmp` package named `snmpconf` which can be used to interactively generate a valid agent configuration.

Note that the `net-snmp-utils` package must be installed in order to use the `snmpwalk` utility described in this section.



Applying the changes

For any changes to the configuration file to take effect, force the `snmpd` service to re-read the configuration by running the following command as root:

```
systemctl reload snmpd.service
```

17.6.3.1. Setting System Information

Net-SNMP provides some rudimentary system information via the `system` tree. For example, the following `snmpwalk` command shows the `system` tree with a default agent configuration.


```
~]# snmpwalk -v2c -c public localhost system
SNMPv2-MIB::sysDescr.0 = STRING: Linux localhost.localdomain 2.6.32-122.el6.x86_64 #1 SMP Wed
Mar 9 23:54:34 EST 2011 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (99554) 0:16:35.54
SNMPv2-MIB::sysContact.0 = STRING: Root <root@localhost> (configure /etc/snmp/
snmp.local.conf)
SNMPv2-MIB::sysName.0 = STRING: localhost.localdomain
SNMPv2-MIB::sysLocation.0 = STRING: Unknown (edit /etc/snmp/snmpd.conf)
```

By default, the `sysName` object is set to the hostname. The `sysLocation` and `sysContact` objects can be configured in the `/etc/snmp/snmpd.conf` file by changing the value of the **`syslocation`** and **`syscontact`** directives, for example:

```
syslocation Datacenter, Row 3, Rack 2
syscontact UNIX Admin <admin@example.com>
```

After making changes to the configuration file, reload the configuration and test it by running the **`snmpwalk`** command again:

```
~]# systemctl reload snmpd.service
~]# snmpwalk -v2c -c public localhost system
SNMPv2-MIB::sysDescr.0 = STRING: Linux localhost.localdomain 2.6.32-122.el6.x86_64 #1 SMP Wed
Mar 9 23:54:34 EST 2011 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (158357) 0:26:23.57
SNMPv2-MIB::sysContact.0 = STRING: UNIX Admin <admin@example.com>
SNMPv2-MIB::sysName.0 = STRING: localhost.localdomain
SNMPv2-MIB::sysLocation.0 = STRING: Datacenter, Row 3, Rack 2
```

17.6.3.2. 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::netSnmAgentOIDs.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 “sysadmin” user with the password “redhatsnmp”:

```
~]# systemctl stop snmpd.service
~]# net-snmp-create-v3-user
Enter a SNMPv3 user name to create:
admin
Enter authentication pass-phrase:
redhatsnmp
Enter encryption pass-phrase:
[press return to reuse the authentication pass-phrase]

adding the following line to /var/lib/net-snmp/snmpd.conf:
    createUser admin MD5 "redhatsnmp" DES
adding the following line to /etc/snmp/snmpd.conf:
    rwuser admin
~]# systemctl start snmpd.service
```

The **rwuser** directive (or **rouser** when the **-ro** command line option is supplied) that **net-snmp-create-v3-user** adds to **/etc/snmp/snmpd.conf** has a similar format to the **rwcommunity** and **rocommunity** directives:

```
directive user [noauth|auth|priv] [OID]
```

... where *user* is a username and *OID* is the SNMP tree to provide access to. By default, the Net-SNMP Agent Daemon allows only authenticated requests (the **auth** option). The **noauth** option allows you to permit unauthenticated requests, and the **priv** option enforces the use of encryption. The **authpriv** option specifies that requests must be authenticated and replies should be encrypted.

For example, the following line grants the user “admin” read-write access to the entire tree:

```
rwuser admin authpriv .1
```

To test the configuration, create a **.snmp** directory in your user's home directory and a configuration file named **snmp.conf** in that directory (**~/ .snmp/snmp.conf**) with the following lines:

```
defVersion 3
defSecurityLevel authPriv
defSecurityName admin
defPassphrase redhatsnmp
```

The **snmpwalk** command will now use these authentication settings when querying the agent:

```
~]$ snmpwalk -v3 localhost system
```



```
SNMPv2-MIB::sysDescr.0 = STRING: Linux localhost.localdomain 2.6.32-122.el6.x86_64 #1 SMP Wed
Mar 9 23:54:34 EST 2011 x86_64
[output truncated]
```

17.6.4. Retrieving Performance Data over SNMP

The Net-SNMP Agent in Fedora 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 17.6.3.2, “Configuring Authentication”](#).

17.6.4.1. 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 17.3, “Available OIDs”](#) summarizes the different OIDs available under that MIB.

Table 17.3. Available OIDs

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

There are also a number of SNMP tables available in the Host Resources MIB which can be used to retrieve a summary of the available information. The following example displays HOST-RESOURCES-MIB::hrFSTable:

```
~]$ 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.

17.6.4.2. 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::memSwapErrorMessage.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
---	---------	------	-------	---	----------	---------

17.6.4.3. 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	AllocationUnits	Size	Used	AllocationFailures	Type	Descr
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

17.6.4.4. Network Information

Information on network devices is provided by the Interfaces MIB. 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	100000000		up

2	eth0	ethernetCsmacd	1500	0 52:54:0:c7:69:58	up
3	eth1	ethernetCsmacd	1500	0 52:54:0:a7:a3:24	down

Network traffic is available under the OIDs IF-MIB::ifOutOctets and IF-MIB::ifInOctets. The following SNMP queries will retrieve network traffic for each of the interfaces on this system:

```
~]$ snmpwalk localhost IF-MIB::ifDescr
IF-MIB::ifDescr.1 = STRING: lo
IF-MIB::ifDescr.2 = STRING: eth0
IF-MIB::ifDescr.3 = STRING: eth1
~]$ snmpwalk localhost IF-MIB::ifOutOctets
IF-MIB::ifOutOctets.1 = Counter32: 10060699
IF-MIB::ifOutOctets.2 = Counter32: 650
IF-MIB::ifOutOctets.3 = Counter32: 0
~]$ snmpwalk localhost IF-MIB::ifInOctets
IF-MIB::ifInOctets.1 = Counter32: 10060699
IF-MIB::ifInOctets.2 = Counter32: 78650
IF-MIB::ifInOctets.3 = Counter32: 0
```

17.6.5. 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 Fedora 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 17.6.3.2, “Configuring Authentication”](#).

17.6.5.1. Extending Net-SNMP with Shell Scripts

The Net-SNMP Agent provides an extension MIB (NET-SNMP-EXTEND-MIB) that can be used to query arbitrary shell scripts. To specify the shell script to run, use the **extend** directive in the **/etc/snmp/snmpd.conf** file. Once defined, the Agent will provide the exit code and any output of the command over SNMP. The example below demonstrates this mechanism with a script which determines the number of **httpd** processes in the process table.

Using the proc directive

The Net-SNMP Agent also provides a built-in mechanism for checking the process table via the **proc** directive. See the **snmpd.conf(5)** manual page for more information.

The exit code of the following shell script is the number of **httpd** processes running on the system at a given point in time:

```
#!/bin/sh
```



```
NUMPIDS=`pgrep httpd | wc -l`  
  
exit $NUMPIDS
```

To make this script available over SNMP, copy the script to a location on the system path, set the executable bit, and add an **extend** directive to the `/etc/snmp/snmpd.conf` file. The format of the **extend** directive is the following:

```
extend name prog args
```

... where *name* is an identifying string for the extension, *prog* is the program to run, and *args* are the arguments to give the program. For instance, if the above shell script is copied to `/usr/local/bin/check_apache.sh`, the following directive will add the script to the SNMP tree:

```
extend httpd_pids /bin/sh /usr/local/bin/check_apache.sh
```

The script can then be queried at `NET-SNMP-EXTEND-MIB::nsExtendObjects`:

```
~]$ snmpwalk localhost NET-SNMP-EXTEND-MIB::nsExtendObjects  
NET-SNMP-EXTEND-MIB::nsExtendNumEntries.0 = INTEGER: 1  
NET-SNMP-EXTEND-MIB::nsExtendCommand."httpd_pids" = STRING: /bin/sh  
NET-SNMP-EXTEND-MIB::nsExtendArgs."httpd_pids" = STRING: /usr/local/bin/check_apache.sh  
NET-SNMP-EXTEND-MIB::nsExtendInput."httpd_pids" = STRING:  
NET-SNMP-EXTEND-MIB::nsExtendCacheTime."httpd_pids" = INTEGER: 5  
NET-SNMP-EXTEND-MIB::nsExtendExecType."httpd_pids" = INTEGER: exec(1)  
NET-SNMP-EXTEND-MIB::nsExtendRunType."httpd_pids" = INTEGER: run-on-read(1)  
NET-SNMP-EXTEND-MIB::nsExtendStorage."httpd_pids" = INTEGER: permanent(4)  
NET-SNMP-EXTEND-MIB::nsExtendStatus."httpd_pids" = INTEGER: active(1)  
NET-SNMP-EXTEND-MIB::nsExtendOutput1Line."httpd_pids" = STRING:  
NET-SNMP-EXTEND-MIB::nsExtendOutputFull."httpd_pids" = STRING:  
NET-SNMP-EXTEND-MIB::nsExtendOutNumLines."httpd_pids" = INTEGER: 1  
NET-SNMP-EXTEND-MIB::nsExtendResult."httpd_pids" = INTEGER: 8  
NET-SNMP-EXTEND-MIB::nsExtendOutLine."httpd_pids".1 = STRING:
```

Note that the exit code ("8" in this example) is provided as an `INTEGER` type and any output is provided as a `STRING` type. To expose multiple metrics as integers, supply different arguments to the script using the **extend** directive. For example, the following shell script can be used to determine the number of processes matching an arbitrary string, and will also output a text string giving the number of processes:

```
#!/bin/sh  
  
PATTERN=$1  
NUMPIDS=`pgrep $PATTERN | wc -l`  
  
echo "There are $NUMPIDS $PATTERN processes."  
exit $NUMPIDS
```

The following `/etc/snmp/snmpd.conf` directives will give both the number of `httpd` PIDs as well as the number of `snmpd` PIDs when the above script is copied to `/usr/local/bin/check_proc.sh`:

```
extend httpd_pids /bin/sh /usr/local/bin/check_proc.sh httpd  
extend snmpd_pids /bin/sh /usr/local/bin/check_proc.sh snmpd
```

The following example shows the output of an **snmpwalk** of the `nsExtendObjects` OID:


```

~]$ snmpwalk localhost NET-SNMP-EXTEND-MIB::nsExtendObjects
NET-SNMP-EXTEND-MIB::nsExtendNumEntries.0 = INTEGER: 2
NET-SNMP-EXTEND-MIB::nsExtendCommand."httpd_pids" = STRING: /bin/sh
NET-SNMP-EXTEND-MIB::nsExtendCommand."snmpd_pids" = STRING: /bin/sh
NET-SNMP-EXTEND-MIB::nsExtendArgs."httpd_pids" = STRING: /usr/local/bin/check_proc.sh httpd
NET-SNMP-EXTEND-MIB::nsExtendArgs."snmpd_pids" = STRING: /usr/local/bin/check_proc.sh snmpd
NET-SNMP-EXTEND-MIB::nsExtendInput."httpd_pids" = STRING:
NET-SNMP-EXTEND-MIB::nsExtendInput."snmpd_pids" = STRING:
...
NET-SNMP-EXTEND-MIB::nsExtendResult."httpd_pids" = INTEGER: 8
NET-SNMP-EXTEND-MIB::nsExtendResult."snmpd_pids" = INTEGER: 1
NET-SNMP-EXTEND-MIB::nsExtendOutLine."httpd_pids".1 = STRING: There are 8 httpd processes.
NET-SNMP-EXTEND-MIB::nsExtendOutLine."snmpd_pids".1 = STRING: There are 1 snmpd processes.

```



Integer exit codes are limited

Integer exit codes are limited to a range of 0–255. For values that are likely to exceed 256, either use the standard output of the script (which will be typed as a string) or a different method of extending the agent.

This last example shows a query for the free memory of the system and the number of `httpd` processes. This query could be used during a performance test to determine the impact of the number of processes on memory pressure:

```

~]$ snmpget localhost \
'NET-SNMP-EXTEND-MIB::nsExtendResult."httpd_pids"' \
UCD-SNMP-MIB::memAvailReal.0
NET-SNMP-EXTEND-MIB::nsExtendResult."httpd_pids" = INTEGER: 8
UCD-SNMP-MIB::memAvailReal.0 = INTEGER: 799664 kB

```

17.6.5.2. 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 Fedora.

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);

```


Obtaining a root OID

The OID **.1.3.6.1.4.1.8072.9999.9999** (NET-SNMP-MIB::netSnmpPlaypen) is typically used for demonstration purposes only. If your organization does not already have a root OID, you can obtain one by contacting your 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
```


17.7. Additional Resources

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

17.7.1. 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.

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 **syslogd**, 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 **syslogd**.

Log files can also be managed by the `journald` daemon – a component of `systemd`. The `journald` daemon captures Syslog messages, kernel log messages, initial RAM disk and early boot messages as well as messages written to standard output and standard error output of all services, indexes them and makes this available to the user. The native journal file format, which is a structured and indexed binary file, improves searching and provides faster operation, and it also stores meta data information like time stamps or user IDs. Log files produced by `journald` are by default not persistent, log files are stored only in memory or a small ring-buffer in the `/run/log/journal/` directory. The amount of logged data depends on free memory, when you reach the capacity limit, the oldest entries are deleted. However, this setting can be altered – see [Section 18.11.5, “Enabling Persistent Storage”](#). For more information on Journal see [Section 18.11, “Using the Journal”](#).

By default, these two logging tools coexist on your system. The `journald` daemon is the primary tool for troubleshooting. It also provides additional data necessary for creating structured log messages. Data acquired by `journald` is forwarded into the `/run/systemd/journal/syslog` socket that may be used by `rsyslogd` to process the data further. However, **rsyslog** does the actual integration by default via the `imjournal` input module, thus avoiding the aforementioned socket. You can also transfer data in the opposite direction, from `rsyslogd` to `journald` with use of `omjournal` module. See [Section 18.7, “Interaction of Rsyslog and Journal”](#) for further information. The integration enables maintaining text-based logs in a consistent format to ensure compatibility with possible applications or configurations dependent on `rsyslogd`. Also, you can maintain `rsyslog` messages in a structured format (see [Section 18.8, “Structured Logging with Rsyslog”](#)).

18.1. 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.

18.2. Basic Configuration of Rsyslog

The main configuration file for **rsyslog** is **/etc/rsyslog.conf**. Here, you can specify *global directives*, *modules*, and *rules* that consist of *filter* and *action* parts. Also, you can add comments in the form of text following a hash sign (#).

18.2.1. Filters

A rule is specified by a *filter* part, which selects a subset of syslog messages, and an *action* part, which specifies what to do with the selected messages. To define a rule in your **/etc/rsyslog.conf** configuration file, define both, a filter and an action, on one line and separate them with one or more spaces or tabs.

rsyslog offers various ways to filter syslog messages according to selected properties. The available filtering methods can be divided into *Facility/Priority-based*, *Property-based*, and *Expression-based* filters.

Facility/Priority-based filters

The most used and well-known way to filter syslog messages is to use the facility/priority-based filters which filter syslog messages based on two conditions: *facility* and *priority* separated by a 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), **ntp** (12), **logaudit** (13), **logalert** (14), **clock** (15), 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 18.1. Facility/Priority-based Filters

The following are a few examples of simple facility/priority-based filters that can be specified in **/etc/rsyslog.conf**. To select all kernel syslog messages with any priority, add the following text into the configuration file:


```
kern.*
```

To select all mail syslog messages with priority **crit** and higher, use this form:

```
mail.crit
```

To select all cron syslog messages except those with the **info** or **debug** priority, set the configuration in the following form:

```
cron.!info,!debug
```

Property-based filters

Property-based filters let you filter syslog messages by any property, such as *timegenerated* or *syslogtag*. For more information on properties, see [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 18.1, “Property-based compare-operations”](#). Both property names and compare operations are case-sensitive.

Property-based filter must start with a colon (:). To define the filter, use the following syntax:

```
:PROPERTY, [!]COMPARE_OPERATION, "STRING"
```

where:

- The *PROPERTY* attribute specifies the desired property.
- The optional exclamation point (!) negates the output of the compare-operation. Other Boolean operators are currently not supported in property-based filters.
- The *COMPARE_OPERATION* attribute specifies one of the compare-operations listed in [Table 18.1, “Property-based compare-operations”](#).
- The *STRING* attribute specifies the value that the text provided by the property is compared to. This value must be enclosed in quotation marks. To escape certain character inside the string (for example a quotation mark (")), use the backslash character (\).

Table 18.1. Property-based compare-operations

Compare-operation	Description
<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> .
<i>regex</i>	Compares the provided POSIX BRE (Basic Regular Expression) against the text provided by the property.

Compare-operation	Description
<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 18.2. Property-based Filters

The following are a few examples of property-based filters that can be specified in `/etc/rsyslog.conf`. To select syslog messages which contain the string **error** in their message text, use:

```
:msg, contains, "error"
```

The following filter selects syslog messages received from the host name **host1**:

```
:hostname, isequal, "host1"
```

To select syslog messages which do not contain any mention of the words **fatal** and **error** with any or no text between them (for example, **fatal lib error**), type:

```
:msg, !regex, "fatal .* error"
```

Expression-based filters

Expression-based filters select syslog messages according to defined arithmetic, Boolean or string operations. Expression-based filters use **rsyslog**'s own scripting language called *RainerScript* to build complex filters.

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 18.3, "Expression-based Filters"](#). The script allows you to use *facility/priority-based* filters inside the expression. On the other hand, *property-based* filters are not recommended here. RainerScript supports regular expressions with specialized functions `re_match()` and `re_extract()`.

Example 18.3. Expression-based Filters

The following expression contains two nested conditions. The log files created by a program called *prog1* are split into two files based on the presence of the "test" string in the message.

```
if $programname == 'prog1' then {
  action(type="omfile" file="/var/log/prog1.log")
  if $msg contains 'test' then
    action(type="omfile" file="/var/log/prog1test.log")
  else
    action(type="omfile" file="/var/log/prog1notest.log")
}
```

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 18.3, “Using the New Configuration Format”](#)

18.2.2. Actions

Actions specify what is to be done with the messages filtered out by an already-defined selector. The following are some of the actions you can define in your rule:

Saving syslog messages to log files

The majority of actions specify to which log file a syslog message is saved. This is done by specifying a file path after your already-defined selector:

```
FILTER PATH
```

where *FILTER* stands for user-specified selector and *PATH* is a path of a target file.

For instance, the following rule is comprised of a selector that selects all **cron** syslog messages and an action that saves them into the **/var/log/cron.log** log file:

```
cron.* /var/log/cron.log
```

By default, the log file is synchronized every time a syslog message is generated. Use a dash mark (-) as a prefix of the file path you specified to omit syncing:

```
FILTER -PATH
```

Note that you might lose information if the system terminates right after a write attempt. However, this setting can 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 18.4. Sending syslog Messages over the Network

The following are some examples of actions that forward syslog messages over the network (note that all actions are preceded with a selector that selects all messages with any priority). To forward messages to 192.168.0.1 via the UDP protocol, type:

```
*.* @192.168.0.1
```

To forward messages to "example.com" using port 18 and the TCP protocol, use:

```
*.* @@example.com:18
```

The following compresses messages with **zlib** (level 9 compression) and forwards them to 2001:db8::1 using the UDP protocol

```
*.* @(z9)[2001:db8::1]
```

Output channels

Output channels are primarily used to specify the maximum size a log file can grow to. This is very useful for log file rotation (for more information see [Section 18.2.5, "Log Rotation"](#)). An output channel is basically a collection of information about the output action. Output channels are defined by the **\$outchannel** directive. To define an output channel in **/etc/rsyslog.conf**, use the following syntax:

```
$outchannel NAME, FILE_NAME, MAX_SIZE, ACTION
```


where:

- The *NAME* attribute specifies the name of the output channel.
- The *FILE_NAME* attribute specifies the name of the output file. Output channels can write only into files, not pipes, terminal, or other kind of output.
- The *MAX_SIZE* attribute represents the maximum size the specified file (in *FILE_NAME*) can grow to. This value is specified in *bytes*.
- The *ACTION* attribute specifies the action that is taken when the maximum size, defined in *MAX_SIZE*, is hit.

To use the defined output channel as an action inside a rule, type:

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

Example 18.5. Output channel log rotation

The following output shows a simple log rotation through the use of an output channel. First, the output channel is defined via the *\$outchannel* directive:

```
$outchannel log_rotation, /var/log/test_log.log, 104857600, /home/joe/
log_rotation_script
```

and then it is used in a rule that selects every syslog message with any priority and executes the previously-defined output channel on the acquired syslog messages:

```
*.* :omfile:$log_rotation
```

Once the limit (in the example 100 MB) is hit, the */home/joe/log_rotation_script* is executed. This script can contain anything from moving the file into a different folder, editing specific content out of it, or simply removing it.

Sending syslog messages to specific users

rsyslog can send syslog messages to specific users by specifying a user name of the user you want to send the messages to (as in [Example 18.7, “Specifying Multiple Actions”](#)). To specify more than one user, separate each user name with a comma (,). To send messages to every user that is currently logged on, use an asterisk (*).

Executing a program

rsyslog lets you execute a program for selected syslog messages and uses the `system()` call to execute the program in shell. To specify a program to be executed, prefix it with a caret character (^). Consequently, specify a template that formats the received message and passes it to the specified executable as a one line parameter (for more information on templates, see [Section 18.2.3, “Templates”](#)).

```
FILTER ^EXECUTABLE; TEMPLATE
```

Here an output of the *FILTER* condition is processed by a program represented by *EXECUTABLE*. This program can be any valid executable. Replace *TEMPLATE* with the name of the formatting template.

Example 18.6. Executing a Program

In the following example, any syslog message with any priority is selected, formatted with the *template* template and passed as a parameter to the **test-program** program, which is then executed with the provided parameter:

```
*.* ^test-program;template
```



Be careful when using the shell execute action

When accepting messages from any host, and using the shell execute action, you may be vulnerable to command injection. An attacker may try to inject and execute 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 18.2.3, “Templates”](#).



Using MySQL and PostgreSQL

Currently, **rsyslog** provides support for MySQL and PostgreSQL databases only. In order to use the MySQL and PostgreSQL database writer functionality, install the *rsyslog-mysql* and *rsyslog-pgsql* packages, respectively. Also, make sure you load the appropriate modules in your **/etc/rsyslog.conf** configuration file:

```
$ModLoad ommysql      # Output module for MySQL support
$ModLoad ompgsql      # Output module for PostgreSQL support
```

For more information on **rsyslog** modules, see [Section 18.6, “Using Rsyslog Modules”](#).

Alternatively, you may use a generic database interface provided by the *omlibdb* module (supports: Firebird/Interbase, MS SQL, Sybase, SQLite, Ingres, Oracle, mSQL).

Discarding syslog messages

To discard your selected messages, use the tilde character (~).

```
FILTER ~
```

The discard action is mostly used to filter out messages before carrying on any further processing. It can be effective if you want to omit some repeating messages that would otherwise fill the log files. The results of discard action depend on where in the configuration file it is specified, for the best results place these actions on top of the actions list. Please note that once a message has been discarded there is no way to retrieve it in later configuration file lines.

For instance, the following rule discards any cron syslog messages:

```
cron.* ~
```

Specifying Multiple Actions

For each selector, you are allowed to specify multiple actions. To specify multiple actions for one selector, write each action on a separate line and precede it with an ampersand (&) character:

```
FILTER ACTION
& ACTION
& ACTION
```

Specifying multiple actions improves the overall performance of the desired outcome since the specified selector has to be evaluated only once.

Example 18.7. Specifying Multiple Actions

In the following example, all kernel syslog messages with the critical priority (*crit*) are sent to user1, processed by the template *temp* and passed on to the *test-program* executable, and forwarded to 192.168.0.1 via the UDP protocol.


```
kern.=crit user1
& ^test-program;temp
& @192.168.0.1
```

Any action can be followed by a template that formats the message. To specify a template, suffix an action with a semicolon (;) and specify the name of the template. For more information on templates, see [Section 18.2.3, “Templates”](#).



Using templates

A template must be defined before it is used in an action, otherwise it is ignored. In other words, template definitions should always precede rule definitions in **/etc/rsyslog.conf**.

18.2.3. Templates

Any output that is generated by **rsyslog** can be modified and formatted according to your needs with the use of *templates*. To create a template use the following syntax in **/etc/rsyslog.conf**:

```
$template TEMPLATE_NAME,"text %PROPERTY% more text", [OPTION]
```

where:

- *\$template* is the template directive that indicates that the text following it, defines a template.
- *TEMPLATE_NAME* is the name of the template. Use this name to refer to the template.
- Anything between the two quotation marks ("...") is the actual template text. Within this text, special characters, such as `\n` for new line or `\r` for carriage return, can be used. Other characters, such as `%` or `"`, have to be escaped if you want to use those characters literally.
- The text specified 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.



The sql and stdsql options

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 18.2.2, “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](http://www.rfc-editor.org/info/rfc3999)¹ date standard.

```
%timegenerated:1:10:date-rfc3339%
```

Template Examples

This section presents a few examples of **rsyslog** templates.

Example 18.8, “A verbose syslog message template” shows a template that formats a syslog message so that it outputs the message's severity, facility, the time stamp of when the message was received, the host name, the message tag, the message text, and ends with a new line.

Example 18.8. A verbose syslog message template

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

Example 18.9, “A wall message template” shows a template that resembles a traditional wall message (a message that is sent 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 18.9. A wall message template

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

Example 18.10, “A database formatted message template” shows a template that formats a syslog message so that it can be used as a database query. Notice the use of the *sql* option at the end of the template specified as the template option. It tells the database writer to format the message as an MySQL SQL query.

Example 18.10. A database formatted message template

```
$template dbFormat, "insert into SystemEvents (Message, Facility, FromHost, Priority,  
DeviceReportedTime, ReceivedAt, InfoUnitID, SysLogTag) values ('%msg%', %syslogfacility  
, '%HOSTNAME%', %syslogpriority%, '%timereported:::date-mysql%', '%timegenerated:::date-  
mysql%', %iut%, '%syslogtag%')", sql
```

rsyslog also contains a set of predefined templates identified by the **RSYSLOG_** prefix. These are reserved for the syslog'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.

¹ <http://www.rfc-editor.org/info/rfc3999>


```
"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%  
\"
```

RSYSLOG_TraditionalForwardFormat

The traditional forwarding format with low-precision time stamps.

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

18.2.4. Global Directives

Global directives are configuration options that apply to the `rsyslogd` daemon. They usually specify a value for a specific predefined variable that affects the behavior of the `rsyslogd` daemon or a rule that follows. All of the global directives must start with a dollar sign (\$). Only one directive can be specified per line. The following is an example of a global directive that specifies the maximum size of the syslog message queue:

```
$MainMsgQueueSize 50000
```

The default size defined for this directive (10,000 messages) can be overridden by specifying a different value (as shown in the example above).

You 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 18.3, “Using the New Configuration Format”](#)). However, classic global directives remain supported as a legacy format.

18.2.5. Log Rotation

The following is a sample `/etc/logrotate.conf` configuration file:

```
# rotate log files weekly
weekly
# keep 4 weeks worth of backlogs
rotate 4
# uncomment this if you want your log files compressed
compress
```

All of the lines in the sample configuration file define global options that apply to every log file. In our example, log files are rotated weekly, rotated log files are kept for four weeks, and all rotated log files are compressed by **gzip** into the **.gz** format. Any lines that begin with a hash sign (#) are comments and are not processed.

You may define configuration options for a specific log file and place it under the global options. However, it is advisable to create a separate configuration file for any specific log file in the `/etc/logrotate.d/` directory and define any configuration options there.

The following is an example of a configuration file placed in the `/etc/logrotate.d/` directory:

```
/var/log/messages {
    rotate 5
    weekly
    postrotate
        /usr/bin/killall -HUP syslogd
    endscript
}
```

The configuration options in this file are specific for the `/var/log/messages` log file only. The settings specified here override the global settings where possible. Thus the rotated `/var/log/messages` log file will be kept for five weeks instead of four weeks as was defined in the global options.

The following is a list of some of the directives you can specify in your **logrotate** configuration file:

- **weekly** — Specifies the rotation of log files to be done weekly. Similar directives include:
 - *daily*
 - *monthly*
 - *yearly*
- **compress** — Enables compression of rotated log files. Similar directives include:
 - *nocompress*

- *compresscmd* — Specifies the command to be used for compressing.
- *uncompresscmd*
- *compressext* — Specifies what extension is to be used for compressing.
- *compressoptions* — 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.

18.3. Using the New Configuration Format

In **rsyslog** version 6, a new configuration syntax was introduced. This new configuration format aims to be more powerful, more intuitive, and to prevent common mistakes by not permitting certain invalid constructs. The syntax enhancement is enabled by the new configuration processor that relies on RainerScript. The legacy format is still fully supported and it is used by default in the **/etc/rsyslog.conf** configuration file.

RainerScript is a scripting language designed for processing network events and configuring event processors such as **rsyslog**. RainerScript was first used to define expression-based filters, see [Example 18.3, “Expression-based Filters”](#). The version of RainerScript in rsyslog version 7 implemented 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”](#).

18.3.1. 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 18.11. Using rulesets

The following rulesets ensure different handling of remote messages coming from different ports. Add the following into `/etc/rsyslog.conf`:

```
ruleset(name="remote-10514") {
    action(type="omfile" file="/var/log/remote-10514")
}

ruleset(name="remote-10515") {
    cron.* action(type="omfile" file="/var/log/remote-10515-cron")
    mail.* action(type="omfile" file="/var/log/remote-10515-mail")
}

input(type="imtcp" port="10514" ruleset="remote-10514");
input(type="imtcp" port="10515" ruleset="remote-10515");
```

Rulesets shown in the above example define log destinations for the remote input from two ports, in case of 10515, messages are sorted according to the facility. Then, the TCP input is enabled and bound to rulesets. Note that you must load the required modules (imtcp) for this configuration to work.

18.3.2. Compatibility with syslogd

From **rsyslog** version 6, compatibility mode specified via the `-c` option has been removed. Also, the syslogd-style command-line options are deprecated and configuring **rsyslog** through these command-line options should be avoided. However, you can use several templates and directives to configure **rsyslogd** to emulate syslogd-like behavior.

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

18.4. Working with Queues in Rsyslog

Queues are used to pass content, mostly syslog messages, between components of **rsyslog**. With queues, **rsyslog** is capable of processing multiple messages simultaneously and to apply several actions to a single message at once. The data flow inside **rsyslog** can be illustrated as follows:

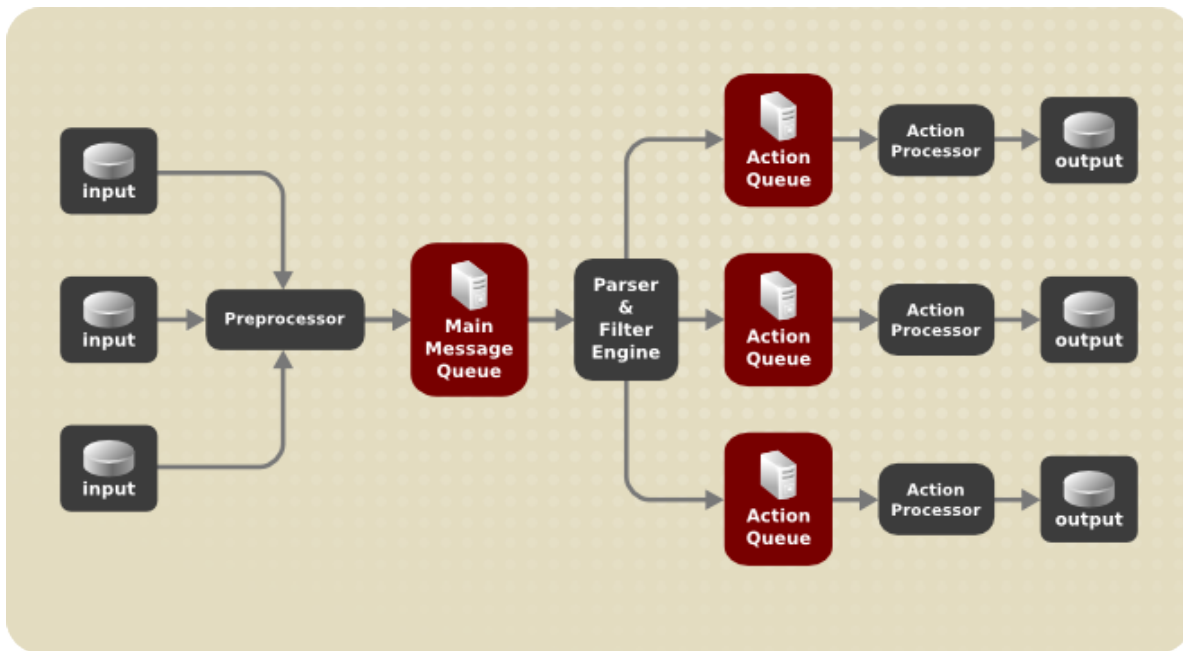


Figure 18.1. Message Flow in Rsyslog

Whenever **rsyslog** receives a message, it passes this message to the preprocessor and then places it into the *main message queue*. Messages wait there to be dequeued and passed to the *rule processor*.

The *rule processor* is a parsing and filtering engine. Here, the rules defined in `/etc/rsyslog.conf` are applied. Based on these rules, the rule processor evaluates which actions are to be performed. Each action has its own action queue. Messages are passed through this queue to the respective action processor which creates the final output. Note that at this point, several actions can run simultaneously on one message. For this purpose, a message is duplicated and passed to multiple action processors.

Only one queue per action is possible. Depending on configuration, the messages can be sent right to the action processor without action queuing. This is the behavior of *direct queues* (see below). In case the output action fails, the action processor notifies the action queue, which then takes an unprocessed element back and after some time interval, the action is attempted again.

To sum up, there are two positions where queues stand in **rsyslog**: either in front of the rule processor as a single *main message queue* or in front of various types of output actions as *action queues*. Queues provide two main advantages that both lead to increased performance of message processing:

- they serve as buffers that *decouple* producers and consumers in the structure of **rsyslog**
- they allow for *parallelization* of actions performed on messages

Apart from this, queues can be configured with several directives to provide optimal performance for your system. These configuration options are covered in the following 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.

18.4.1. 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:

```
$ObjectQueueFilename name
```

This sets a *name* prefix for the file followed by a 7-digit number starting at one and incremented for each file.

In-memory Queues

With in-memory queue, the enqueued messages are held in memory which makes the process very fast. The queued data 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:

```
$ObjectQueueType LinkedList
```

```
$ObjectQueueType FixedArray
```

Replace *object* with **MainMsg** or with **Action** to use this option to the main message queue or for an action queue respectively.

Disk-Assisted In-memory Queues

Both disk and in-memory queues have their advantages and **rsyslog** lets you combine them in *disk-assisted in-memory queues*. To do so, configure a normal in-memory queue and then add the **\$ObjectQueueFileName** directive to define a file name for disk assistance. This queue then becomes *disk-assisted*, which means it couples an in-memory queue with a disk queue to work in tandem.

The disk queue is activated if the in-memory queue is full or needs to persist after shutdown. With a disk-assisted queue, you can set both disk-specific and in-memory specific configuration parameters. This type of queue is probably the most commonly used, it is especially useful for potentially long-running and unreliable actions.

To specify the functioning of a disk-assisted in-memory queue, use the so-called *watermarks*:

```
$ObjectQueueHighWatermark number
```

```
$ObjectQueueLowWatermark number
```


Replace *object* with **MainMsg** or with **Action** to use this option to the main message queue or for an action queue respectively. Replace *number* with a number of enqueued messages. When an in-memory queue reaches the number defined by the high watermark, it starts writing messages to disk and continues until the in-memory queue size drops to the number defined with the low watermark. Correctly set watermarks minimize unnecessary disk writes, but also leave memory space for message bursts since writing to disk files is rather lengthy. Therefore, the high watermark must be lower than the whole queue capacity set with *\$objectQueueSize*. The difference between the high watermark and the overall queue size is a spare memory buffer reserved for message bursts. On the other hand, setting the high watermark too low will turn on disk assistance unnecessarily often.

Example 18.12. 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 18.6.4, “Using RELP”](#).

Procedure 18.1. 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:

1. Create a working directory to store the queue files. For example:

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

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

```
$WorkDirectory /rsyslog/work

$ActionQueueType LinkedList
$ActionQueueFileName example_fwd
$ActionResumeRetryCount -1
$ActionQueueSaveOnShutdown on
*. *                @@example.com:18
```

Where:

- the **/rsyslog/work/** directory created in the previous step is marked as a working directory,
- **\$ActionQueueType** enables a LinkedList in-memory queue,
- **\$ActionFileName** defines a disk storage, in this case the backup files are created in the **/rsyslog/work/** 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 18.2. Forwarding To Multiple Servers

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

1. Create a working directory for rsyslog to store the queue files. For example:

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

2. 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:

```
$WorkDirectory /rsyslog/work

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

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

18.4.2. Managing Queues

All types of queues can be further configured to match your requirements. You can use several directives to modify both action queues and the main message queue. Currently, there are more than 20 queue parameters available, see [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:

```
$objectQueueHighWatermark number
```

Replace *object* with **MainMsg** or with **Action** to use this option to the main message queue or for an action queue respectively. Replace *number* with a number of enqueued messages. You can set the queue size only as the number of messages, not as their actual memory size. The default queue size is 10,000 messages for the main message queue and ruleset queues, and 1000 for action queues.

Disk assisted queues are unlimited by default and can not be restricted with this directive, but you can reserve them physical disk space in bytes with the following settings:

```
$objectQueueMaxDiscSpace number
```


Replace *object* with **MainMsg** or with **Action**. When the size limit specified by *number* is hit, messages are discarded until sufficient amount of space is freed by dequeued messages.

Discarding Messages

When a queue reaches a certain number of messages, you can discard less important messages in order to save space in the queue for entries of higher priority. The threshold that launches the discarding process can be set with the so-called *discard mark*:

```
$objectQueueDiscardMark number
```

Replace *object* with **MainMsg** or with **Action** to use this option to the main message queue or for an action queue respectively. Here, *number* stands for a number of messages that have to be in the queue to start the discarding process. To define which messages to discard, use:

```
$objectQueueDiscardSeverity priority
```

Replace *priority* with one of the following keywords (or with a number): **debug** (7), **info** (6), **notice** (5), **warning** (4), **err** (3), **crit** (2), **alert** (1), and **emerg** (0). With this setting, both newly incoming and already queued messages with lower than defined priority are erased from the queue immediately after the discard mark is reached.

Using Timeframes

You can configure **rsyslog** to process queues during a specific time period. With this option you can, for example, transfer some processing into off-peak hours. To define a time frame, use the following syntax:

```
$objectQueueDequeueTimeBegin hour
```

```
$objectQueueDequeueTimeEnd hour
```

With *hour* you can specify hours that bound your time frame. Use the 24-hour format without minutes.

Configuring Worker Threads

A *worker thread* performs a specified action on the enqueued message. For example, in the main message queue, a worker task is to apply filter logic to each incoming message and enqueue them to the relevant action queues. When a message arrives, a worker thread is started automatically. When the number of messages reaches a certain number, another worker thread is turned on. To specify this number, use:

```
$objectQueueWorkerThreadMinimumMessages number
```

Replace *number* with a number of messages that will trigger a supplemental worker thread. For example, with *number* set to 100, a new worker thread is started when more than 100 messages arrive. When more than 200 messages arrive, the third worker thread starts and so on. However, too many working threads running in parallel becomes ineffective, so you can limit the maximum number of them by using:

```
$objectQueueWorkerThreads number
```


where *number* stands for a maximum number 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:

```
$objectQueueWorkerTimeoutThreadShutdown time
```

Replace *time* with the duration set in milliseconds. Without this setting, a zero timeout is applied and a worker thread is terminated immediately when it runs out of messages. If you specify *time* as **-1**, no thread will be closed.

Batch Dequeueing

To increase performance, you can configure **rsyslog** to dequeue multiple messages at once. To set the upper limit for such dequeueing, use:

```
$objectQueueDequeueBatchSize 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:

```
$objectQueueTimeoutShutdown time
```

Specify *time* in milliseconds. If after that period there are still some enqueued messages, workers finish the current data element and then terminate. Unprocessed messages are therefore lost. Another time interval can be set for workers to finish the final element:

```
$objectQueueTimeoutActionCompletion time
```

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

```
$objectQueueTimeoutSaveOnShutdown time
```

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

18.5. 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 18.12, “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 Fedora 26. If required, to ensure that it is, enter the following command as **root**:

```
~]# dnf install rsyslog
```


The steps in this procedure must be followed on the system that you intend to use as your logging server. All steps in this procedure must be made as the root user:

1. Configure the firewall to allow rsyslog TCP traffic.

- The default port for rsyslog TCP traffic is **514**. To allow TCP traffic on this port, enter a command as follows:

```
~]# firewall-cmd --zone=zone --add-port=514/tcp
success
```

Where *zone* is the zone of the interface to use.

2. 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 514
```

Save the changes to the **/etc/rsyslog.conf** file.

3. The rsyslog service must be running on both the logging server and the systems attempting to log to it.

- a. Use the **systemctl** command to start the rsyslog service.

```
~]# systemctl start rsyslog
```

- b. To ensure the rsyslog service starts automatically in future, enter the following command as root:

```
~]# systemctl enable rsyslog
```

Your log server is now configured to receive and store log files from the other systems in your environment.

18.5.1. 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="514" ruleset="remote1")
```

18.6. 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** and **imjournal**.
- **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 18.2.3, “Templates”](#). The name of a string generator module always starts with the **sm** prefix, such as **smfile** or **smtradfile**.
- **Library Modules** — Library modules provide functionality for other loadable modules. These modules are loaded automatically by **rsyslog** when needed and cannot be configured by the user.

A comprehensive list of all available modules and their detailed description can be found at http://www.rsyslog.com/doc/rsyslog_conf_modules.html².



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.

² http://www.rsyslog.com/doc/rsyslog_conf_modules.html/

18.6.1. 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 18.2.1, “Filters”](#).

Example 18.13. Importing Text Files

The Apache HTTP server creates log files in text format. To apply the processing capabilities of **rsyslog** to apache error messages, first use the **imfile** module to import the messages. Add the following into **/etc/rsyslog.conf**:

```
$ModLoad imfile

$InputFileName /var/log/httpd/error_log
```



```
$InputFileTag apache-error:
$InputFileStateFile state-apache-error
$InputRunFileMonitor
```

18.6.2. 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 18.14. Exporting Rsyslog Messages to a Database

To store the rsyslog messages in a MySQL database, add the following into `/etc/rsyslog.conf`:

```
$ModLoad ommysql

$ActionOmmysqlServerPort 1234
*. * :ommysql:database-server,database-name,database-userid,database-password
```

First, the output module is loaded, then the communication port is specified. Additional information, such as name of the server and the database, and authentication data, is specified on the last line of the above example.

18.6.3. Enabling Encrypted Transport

Confidentiality and integrity in network transmissions can be provided by either the *TLS* or *GSSAPI* encryption protocol.

Transport Layer Security (TLS) is a cryptographic protocol designed to provide communication security over the network. When using TLS, rsyslog messages are encrypted before sending, and mutual authentication exists between the sender and receiver.

Generic Security Service API (GSSAPI) is an application programming interface for programs to access security services. To use it in connection with **rsyslog** you must have a functioning **Kerberos** environment.

18.6.4. 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.

18.7. Interaction of Rsyslog and Journal

As mentioned above, **Rsyslog** and **Journal**, the two logging applications present on your system, have several distinctive features that make them suitable for specific use cases. In many situations it is useful to combine their capabilities, for example to create structured messages and store them in a file database (see [Section 18.8, “Structured Logging with Rsyslog”](#)). A communication interface needed for this cooperation is provided by input and output modules on the side of **Rsyslog** and by the **Journal**'s communication socket.

By default, `rsyslogd` uses the `imjournal` module as a default input mode for journal files. With this module, you import not only the messages but also the structured data provided

by `journald`. Also, older data can be imported from `journald` (unless forbidden with the **\$ImjournalIgnorePreviousMessages** directive). See [Section 18.8.1, “Importing Data from Journal”](#) for basic configuration of `imjournal`.

As an alternative, configure `rsyslogd` to read from the socket provided by `journal` as an output for syslog-based applications. The path to the socket is **/run/systemd/journal/syslog**. Use this option when you want to maintain plain rsyslog messages. Compared to `imjournal` the socket input currently offers more features, such as ruleset binding or filtering. To import **Journal** data through the socket, use the following configuration in **/etc/rsyslog.conf**:

```
$ModLoad imuxsock
$OmitLocalLogging off
```

The above syntax loads the `imuxsock` module and turns off the **\$OmitLocalLogging** directive, which enables the import through the system socket. The path to this socket is specified separately in **/etc/rsyslog.d/listen.conf** as follows:

```
$SystemLogSocketName /run/systemd/journal/syslog
```

You can also output messages from **Rsyslog** to **Journal** with the `omjournal` module. Configure the output in **/etc/rsyslog.conf** as follows:

```
$ModLoad omjournal
*. * :omjournal:
```

For instance, the following configuration forwards all received messages on tcp port 10514 to the Journal:

```
$ModLoad imtcp
$ModLoad omjournal

$RuleSet remote
*. * :omjournal:

$InputTCPServerBindRuleset remote
$InputTCPServerRun 10514
```

18.8. Structured Logging with Rsyslog

On systems that produce large amounts of log data, it can be convenient to maintain log messages in a *structured format*. With structured messages, it is easier to search for particular information, to produce statistics and to cope with changes and inconsistencies in message structure. **Rsyslog** uses the **JSON** (JavaScript Object Notation) format to provide structure for log messages.

Compare the following unstructured log message:

```
Oct 25 10:20:37 localhost anacron[1395]: Jobs will be executed sequentially
```

with a structured one:


```
{ "timestamp": "2013-10-25T10:20:37", "host": "localhost", "program": "anacron", "pid": "1395",
  "msg": "Jobs will be executed sequentially" }
```

Searching structured data with use of key-value pairs is faster and more precise than searching text files with regular expressions. The structure also lets you to search for the same entry in messages produced by various applications. Also, JSON files can be stored in a document database such as MongoDB, which provides additional performance and analysis capabilities. On the other hand, a structured message requires more disk space than the unstructured one.

In **rsyslog**, log messages with meta data are pulled from **Journal** with use of the `imjournal` module. With the `mmjsonparse` module, you can parse data imported from **Journal** and from other sources and process them further, for example as a database output. For parsing to be successful, `mmjsonparse` requires input messages to be structured in a way that is defined by the **Lumberjack** project.

The **Lumberjack** project aims to add structured logging to **rsyslog** in a backward-compatible way. To identify a structured message, **Lumberjack** specifies the `@cee:` string that prepends the actual JSON structure. Also, **Lumberjack** defines the list of standard field names that should be used for entities in the JSON string. For more information on **Lumberjack**, see [the section called “Online Documentation”](#).

The following is an example of a lumberjack-formatted message:

```
@cee: { "pid": 17055, "uid": 1000, "gid": 1000, "appname": "logger", "msg": "Message text." }
```

To build this structure inside **Rsyslog**, a template is used, see [Section 18.8.2, “Filtering Structured Messages”](#). Applications and servers can employ the `libumberlog` library to generate messages in the lumberjack-compliant form. For more information on `libumberlog`, see [the section called “Online Documentation”](#).

18.8.1. Importing Data from Journal

The `imjournal` module is **Rsyslog**'s input module to natively read the journal files (see [Section 18.7, “Interaction of Rsyslog and Journal”](#)). Journal messages are then logged in text format as other **rsyslog** messages. However, with further processing, it is possible to translate meta data provided by **Journal** into a structured message.

To import data from **Journal** to **Rsyslog**, use the following configuration in `/etc/rsyslog.conf`:

```
$ModLoad imjournal

$imjournalPersistStateInterval number_of_messages
$imjournalStateFile path
$imjournalRateLimitInterval seconds
$imjournalRateLimitBurst burst_number
$imjournalIgnorePreviousMessages off/on
```

- With *number_of_messages*, you can specify how often the journal data must be saved. This will happen each time the specified number of messages is reached.
- Replace *path* with a path to the state file. This file tracks the journal entry that was the last one processed.
- With *seconds*, you set the length of the rate limit interval. The number of messages processed during this interval can not exceed the value specified in *burst_number*. The default setting is

20,000 messages per 600 seconds. Rsyslog discards messages that come after the maximum burst within the time frame specified.

- With **\$ImjournalIgnorePreviousMessages** you can ignore messages that are currently in Journal and import only new messages, which is used when there is no state file specified. The default setting is **off**. Please note that if this setting is off and there is no state file, all messages in the Journal are processed, even if they were already processed in a previous rsyslog session.

Note

You can use `imjournal` simultaneously with `imuxsock` module that is the traditional system log input. However, to avoid message duplication, you must prevent `imuxsock` from reading the Journal's system socket. To do so, use the **\$OmitLocalLogging** directive:

```
$ModLoad imuxsock
$ModLoad imjournal

$OmitLocalLogging on
$AddUnixListenSocket /run/systemd/journal/syslog
```

You can translate all data and meta data stored by **Journal** into structured messages. Some of these meta data entries are listed in [Example 18.16, “Verbose journalctl Output”](#), for a complete list of journal fields see the **systemd.journal-fields(7)** manual page. For example, it is possible to focus on *kernel journal fields*, that are used by messages originating in the kernel.

18.8.2. Filtering Structured Messages

To create a lumberjack-formatted message that is required by **rsyslog**'s parsing module, use the following template:

```
template(name="CEETemplate" type="string" string="%TIMESTAMP% %HOSTNAME% %syslogtag% @cee: %
$!all-json%\n")
```

This template prepends the **@cee:** string to the JSON string and can be applied, for example, when creating an output file with `omfile` module. To access JSON field names, use the **\$!** prefix. For example, the following filter condition searches for messages with specific *hostname* and *UID*:

```
($!hostname == "hostname" && $!UID== "UID")
```

18.8.3. Parsing JSON

The `mmjsonparse` module is used for parsing structured messages. These messages can come from **Journal** or from other input sources, and must be formatted in a way defined by the **Lumberjack** project. These messages are identified by the presence of the **@cee:** string. Then, `mmjsonparse` checks if the JSON structure is valid and then the message is parsed.

To parse lumberjack-formatted JSON messages with `mmjsonparse`, use the following configuration in the **/etc/rsyslog.conf**:


```
$ModLoad mmjsonparse
*. * :mmjsonparse:
```

In this example, the `mmjsonparse` module is loaded on the first line, then all messages are forwarded to it. Currently, there are no configuration parameters available for `mmjsonparse`.

18.8.4. Storing Messages in the MongoDB

Rsyslog supports storing JSON logs in the MongoDB document database through the *ommongodb* output module.

To forward log messages into MongoDB, use the following syntax in the `/etc/rsyslog.conf` (configuration parameters for *ommongodb* are available only in the new configuration format; see [Section 18.3, “Using the New Configuration Format”](#)):

```
$ModLoad ommongodb

*. * action(type="ommongodb" server="DB_server" serverport="port" db="DB_name"
collection="collection_name" uid="UID" pwd="password")
```

- Replace *DB_server* with the name or address of the MongoDB server. Specify *port* to select a non-standard port from the MongoDB server. The default *port* value is `0` and usually there is no need to change this parameter.
- With *DB_name*, you identify to which database on the MongoDB server you want to direct the output. Replace *collection_name* with the name of a collection in this database. In MongoDB, collection is a group of documents, the equivalent of an RDBMS table.
- You can set your login details by replacing *UID* and *password*.

You can shape the form of the final database output with use of templates. By default, **rsyslog** uses a template based on standard **lumberjack** field names.

18.9. Debugging Rsyslog

To run `rsyslogd` in debugging mode, use the following command:

```
rsyslogd -dn
```

With this command, `rsyslogd` produces debugging information and prints it to the standard output. The `-n` stands for "no fork". You can modify debugging with environmental variables, for example, you can store the debug output in a log file. Before starting `rsyslogd`, type the following on the command line:

```
export RSYSLOG_DEBUGLOG="path"
export RSYSLOG_DEBUG="Debug"
```

Replace *path* with a desired location for the file where the debugging information will be logged. For a complete list of options available for the `RSYSLOG_DEBUG` variable, see the related section in the **rsyslogd(8)** manual page.

To check if syntax used in the `/etc/rsyslog.conf` file is valid use:

```
rsyslogd -N 1
```

Where **1** represents level of verbosity of the output message. This is a forward compatibility option because currently, only one level is provided. However, you must add this argument to run the validation.

18.10. Troubleshooting Logging to a Server

- Ensure the time is correctly set on the systems generating the log messages as well as on any logging servers. See [Chapter 3, Configuring the Date and Time](#) for information on checking and setting the time. See [Chapter 15, Configuring NTP Using ntpd](#) and [Chapter 14, Configuring NTP Using the chrony Suite](#) for information on using NTP to keep the system clock accurately set.
- On a logging server, check that the firewall has the appropriate ports open to allow ingress of either UDP or TCP, depending on what traffic and port the sending systems are configured to use. For example:

```
~]# firewall-cmd --zone=public --list-ports
```

For more information on opening and closing ports in `firewalld`, see the [Red Hat Enterprise Linux 7 Security Guide](#)³. Review the configuration of the logging server to ensure it is listening on the same port the sending systems are configured to send on, and all are set to use the same protocol.

- Use the **logger** command to generate test log messages. For example:

```
~]$ logger -p authpriv.info "Test Secret"  
~]$ logger -p auth.info "Test Info"
```

See the **logger(1)** manual page for more information on the **logger** command.

18.11. Using the Journal

The Journal is a component of **systemd** that is responsible for viewing and management of log files. It can be used in parallel, or in place of a traditional syslog daemon, such as `rsyslogd`. The Journal was developed to address problems connected with traditional logging. It is closely integrated with the rest of the system, supports various logging technologies and access management for the log files.

Logging data is collected, stored, and processed by the Journal's `journald` service. It creates and maintains binary files called *journals* based on logging information that is received from the kernel, from user processes, from standard output, and standard error output of system services or via its native API. These journals are structured and indexed, which provides relatively fast seek times. Journal entries can carry a unique identifier. The `journald` service collects numerous meta data fields for each log message. The actual journal files are secured, and therefore cannot be manually edited.

18.11.1. Viewing Log Files

To access the journal logs, use the **journalctl** tool. For a basic view of the logs type as root:

³ https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/

journalctl

An output of this command is a list of all log files generated on the system including messages generated by system components and by users. The structure of this output is similar to one used in `/var/log/messages/` but with certain improvements:

- the priority of entries is marked visually. Lines of error priority and higher are highlighted with red color and a bold font is used for lines with notice and warning priority
- the time stamps are converted for the local time zone of your system
- all logged data is shown, including rotated logs
- the beginning of a boot is tagged with a special line

Example 18.15. Example Output of journalctl

The following is an example output provided by the **journalctl** tool. When called without parameters, the listed entries begin with a time stamp, then the host name and application that performed the operation is mentioned followed by the actual message. This example shows the first three entries in the journal log:

```
# journalctl
-- Logs begin at Thu 2013-08-01 15:42:12 CEST, end at Thu 2013-08-01 15:48:48 CEST. --
Aug 01 15:42:12 localhost systemd-journal[54]: Allowing runtime journal files to grow to
49.7M.
Aug 01 15:42:12 localhost kernel: Initializing cgroup subsys cpuset
Aug 01 15:42:12 localhost kernel: Initializing cgroup subsys cpu

[...]
```

In many cases, only the latest entries in the journal log are relevant. The simplest way to reduce **journalctl** output is to use the **-n** option that lists only the specified number of most recent log entries:

```
journalctl -n Number
```

Replace *Number* with the number of lines to be shown. When no number is specified, **journalctl** displays the ten most recent entries.

The **journalctl** command allows controlling the form of the output with the following syntax:

```
journalctl -o form
```

Replace *form* with a keyword specifying a desired form of output. There are several options, such as **verbose**, which returns full-structured entry items with all fields, **export**, which creates a binary stream suitable for backups and network transfer, and **json**, which formats entries as JSON data structures. For the full list of keywords, see the **journalctl(1)** manual page.

Example 18.16. Verbose journalctl Output

To view full meta data about all entries, type:

```
# journalctl -o verbose
```



```
[...]

Fri 2013-08-02 14:41:22 CEST
[s=e1021ca1b81e4fc688fad6a3ea21d35b;i=55c;b=78c81449c920439da57da7bd5c56a770;m=27cc
  _BOOT_ID=78c81449c920439da57da7bd5c56a770
  _PRIORITY=5
  _SYSLOG_FACILITY=3
  _TRANSPORT=syslog
  _MACHINE_ID=69d27b356a94476da859461d3a3bc6fd
  _HOSTNAME=localhost.localdomain
  _PID=562
  _COMM=dbus-daemon
  _EXE=/usr/bin/dbus-daemon
  _CMDLINE=/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --
systemd-activation
  _SYSTEMD_CGROUP=/system/dbus.service
  _SYSTEMD_UNIT=dbus.service
  _SYSLOG_IDENTIFIER=dbus
  _SYSLOG_PID=562
  _UID=81
  _GID=81
  _SELINUX_CONTEXT=system_u:system_r:system_dbusd_t:s0-s0:c0.c1023
  MESSAGE=[system] Successfully activated service 'net.reactivated.Fprint'
  _SOURCE_REALTIME_TIMESTAMP=1375447282839181

[...]
```

This example lists fields that identify a single log entry. These meta data can be used for message filtering as shown in [the section called “Advanced Filtering”](#). For a complete description of all possible fields see the **systemd.journal-fields(7)** manual page.

18.11.2. Access Control

By default, **Journal** users without root privileges can only see log files generated by them. The system administrator can add selected users to the *adm* group, which grants them access to complete log files. To do so, type as root:

```
usermod -a -G adm username
```

Here, replace *username* with a name of the user to be added to the *adm* group. This user then receives the same output of the **journalctl** command as the root user. Note that access control only works when persistent storage is enabled for **Journal**.

18.11.3. Using The Live View

When called without parameters, **journalctl** shows the full list of entries, starting with the oldest entry collected. With the live view, you can supervise the log messages in real time as new entries are continuously printed as they appear. To start **journalctl** in live view mode, type:

```
journalctl -f
```

This command returns a list of the ten most current log lines. The **journalctl** utility then stays running and waits for new changes to show them immediately.

18.11.4. Filtering Messages

The output of the **journalctl** command executed without parameters is often extensive, therefore you can use various filtering methods to extract information to meet your needs.

Filtering by Priority

Log messages are often used to track erroneous behavior on the system. To view only entries with a selected or higher priority, use the following syntax:

```
journalctl -p priority
```

Here, replace *priority* with one of the following keywords (or with a number): **debug** (7), **info** (6), **notice** (5), **warning** (4), **err** (3), **crit** (2), **alert** (1), and **emerg** (0).

Example 18.17. Filtering by Priority

To view only entries with *error* or higher priority, use:

```
journalctl -p err
```

Filtering by Time

To view log entries only from the current boot, type:

```
journalctl -b
```

If you reboot your system just occasionally, the **-b** will not significantly reduce the output of **journalctl**. In such cases, time-based filtering is more helpful:

```
journalctl --since=value --until=value
```

With **--since** and **--until**, you can view only log messages created within a specified time range. You can pass *values* to these options in form of date or time or both as shown in the following example.

Example 18.18. Filtering by Time and Priority

Filtering options can be combined to reduce the set of results according to specific requests. For example, to view the *warning* or higher priority messages from a certain point in time, type:

```
journalctl -p warning --since="2013-3-16 23:59:59"
```

Advanced Filtering

[Example 18.16, “Verbose journalctl Output”](#) lists a set of fields that specify a log entry and can all be used for filtering. For a complete description of meta data that **systemd** can store, see the **systemd.journal-fields(7)** manual page. This meta data is collected for each log message, without user intervention. Values are usually text-based, but can take binary and large values; fields can have multiple values assigned though it is not very common.

To view a list of unique values that occur in a specified field, use the following syntax:

```
journalctl -F fieldname
```

Replace *fieldname* with a name of a field you are interested in.

To show only log entries that fit a specific condition, use the following syntax:


```
journalctl fieldname=value
```

Replace *fieldname* with a name of a field and *value* with a specific value contained in that field. As a result, only lines that match this condition are returned.

Tab Completion on Field Names

As the number of meta data fields stored by systemd is quite large, it is easy to forget the exact name of the field of interest. When unsure, type:

```
journalctl
```

and press the **Tab** key two times. This shows a list of available field names. **Tab** completion based on context works on field names, so you can type a distinctive set of letters from a field name and then press **Tab** to complete the name automatically. Similarly, you can list unique values from a field. Type:

```
journalctl fieldname=
```

and press **Tab** two times. This serves as an alternative to `journalctl -F fieldname`.

You can specify multiple values for one field:

```
journalctl fieldname=value1 fieldname=value2 ...
```

Specifying two matches for the same field results in a logical OR combination of the matches. Entries matching *value1* or *value2* are displayed.

Also, you can specify multiple field-value pairs to further reduce the output set:

```
journalctl fieldname1=value fieldname2=value ...
```

If two matches for different field names are specified, they will be combined with a logical AND. Entries have to match both conditions to be shown.

With use of the + symbol, you can set a logical OR combination of matches for multiple fields:

```
journalctl fieldname1=value + fieldname2=value ...
```

This command returns entries that match at least one of the conditions, not only those that match both of them.

Example 18.19. Advanced filtering

To display entries created by `avahi-daemon.service` or `crond.service` under user with UID 70, use the following command:

```
journalctl _UID=70 _SYSTEMD_UNIT=avahi-daemon.service _SYSTEMD_UNIT=crond.service
```


Since there are two values set for the `_SYSTEMD_UNIT` field, both results will be displayed, but only when matching the `_UID=70` condition. This can be expressed simply as: `(UID=70 and (avahi or cron))`.

You can apply the aforementioned filtering also in the live-view mode to keep track of the latest changes in the selected group of log entries:

```
journalctl -f fieldname=value ...
```

18.11.5. Enabling Persistent Storage

By default, **Journal** stores log files only in memory or a small ring-buffer in the `/run/log/journal/` directory. This is sufficient to show recent log history with **journalctl**. This directory is volatile, log data is not saved permanently. With the default configuration, **syslog** reads the journal logs and stores them in the `/var/log/` directory. With persistent logging enabled, journal files are stored in `/var/log/journal` which means they persist after reboot. Journal can then replace **rsyslog** for some users (but see the chapter introduction).

Enabled persistent storage has the following advantages

- Richer data is recorded for troubleshooting in a longer period of time
- For immediate troubleshooting, richer data is available after a reboot
- Server console currently reads data from journal, not log files

Persistent storage has also certain disadvantages:

- Even with persistent storage the amount of data stored depends on free memory, there is no guarantee to cover a specific time span
- More disk space is needed for logs

To enable persistent storage for Journal, create the journal directory manually as shown in the following example. As root type:

```
mkdir -p /var/log/journal
```

Then, restart `journald` to apply the change:

```
systemctl restart systemd-journald
```

18.12. Managing Log Files in a Graphical Environment

As an alternative to the aforementioned command-line utilities, Red Hat Enterprise Linux 7 provides an accessible GUI for managing log messages.

18.12.1. 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 **System Log**.

Installing the `gnome-system-log` package

In order to use the **System Log**, first ensure the `gnome-system-log` package is installed on your system by running, as root:

```
~]# dnf install gnome-system-log
```

For more information on installing packages with DNF, see [Section 6.2.4, “Installing Packages”](#).

After you have installed the `gnome-system-log` package, open the **System Log** by clicking **Applications** → **System Tools** → **System Log**, or type the following command at a shell prompt:

```
~]$ gnome-system-log
```

The application only displays log files that exist; thus, the list might differ from the one shown in [Figure 18.2, “System Log”](#).

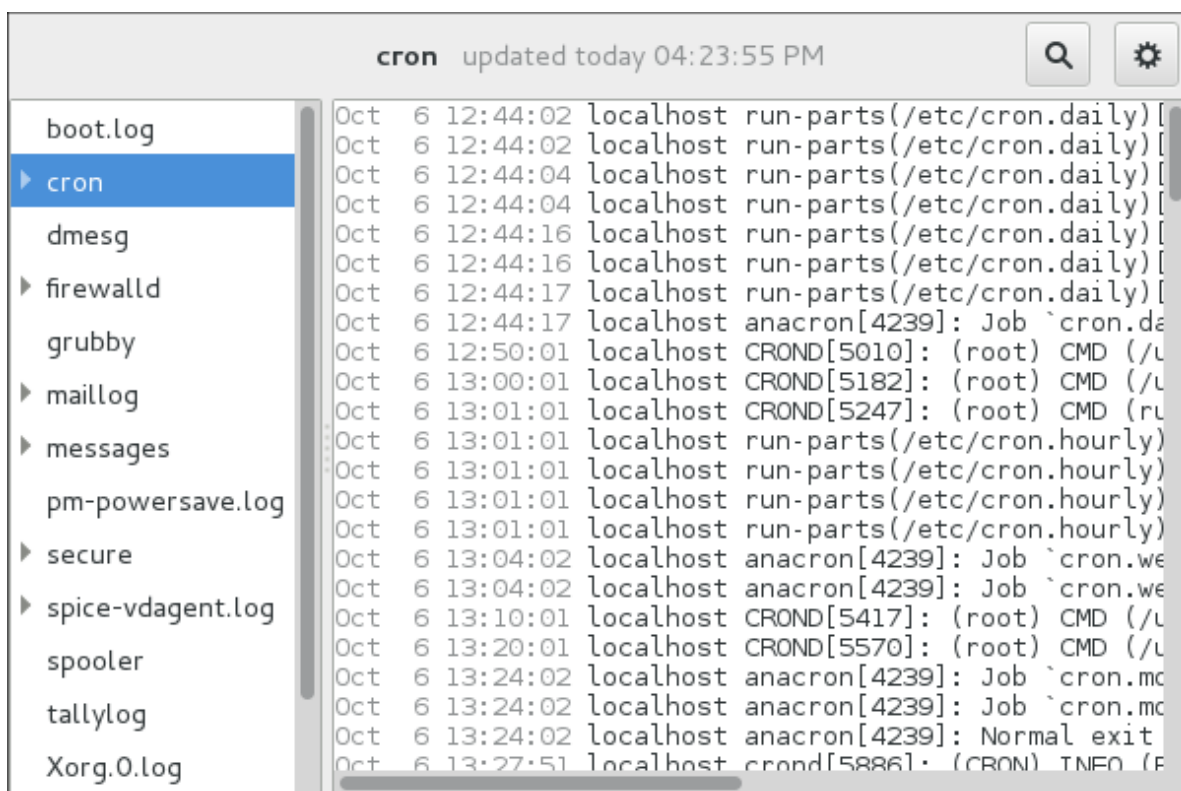


Figure 18.2. System Log

The **System Log** application lets you filter any existing log file. Click on the button marked with the gear symbol to view the menu, select **Filters** → **Manage Filters** to define or edit the desired filter.

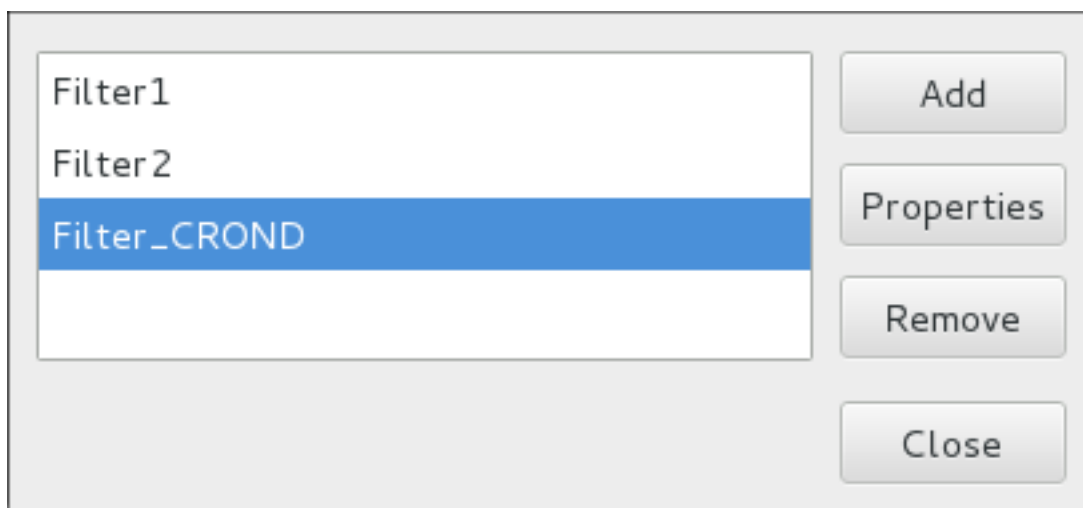


Figure 18.3. System Log - Filters

Adding or editing a filter lets you define its parameters as is shown in [Figure 18.4](#), “System Log - defining a filter”.

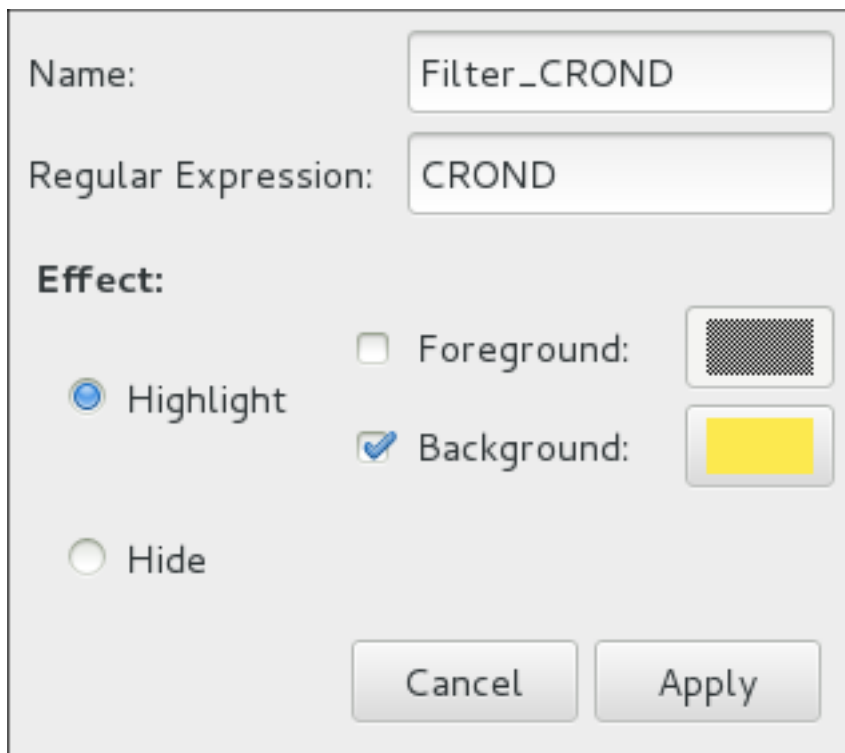


Figure 18.4. System Log - defining a filter

When defining a filter, the following parameters can be edited:

- **Name** — Specifies the name of the filter.
- **Regular Expression** — Specifies the regular expression that will be applied to the log file and will attempt to match any possible strings of text in it.
- **Effect**
 - **Highlight** — If checked, the found results will be highlighted with the selected color. You may select whether to highlight the background or the foreground of the text.

- **Hide** — If checked, the found results will be hidden from the log file you are viewing.

When you have at least one filter defined, it can be selected from the **Filters** menu and it will automatically search for the strings you have defined in the filter and highlight or hide every successful match in the log file you are currently viewing.

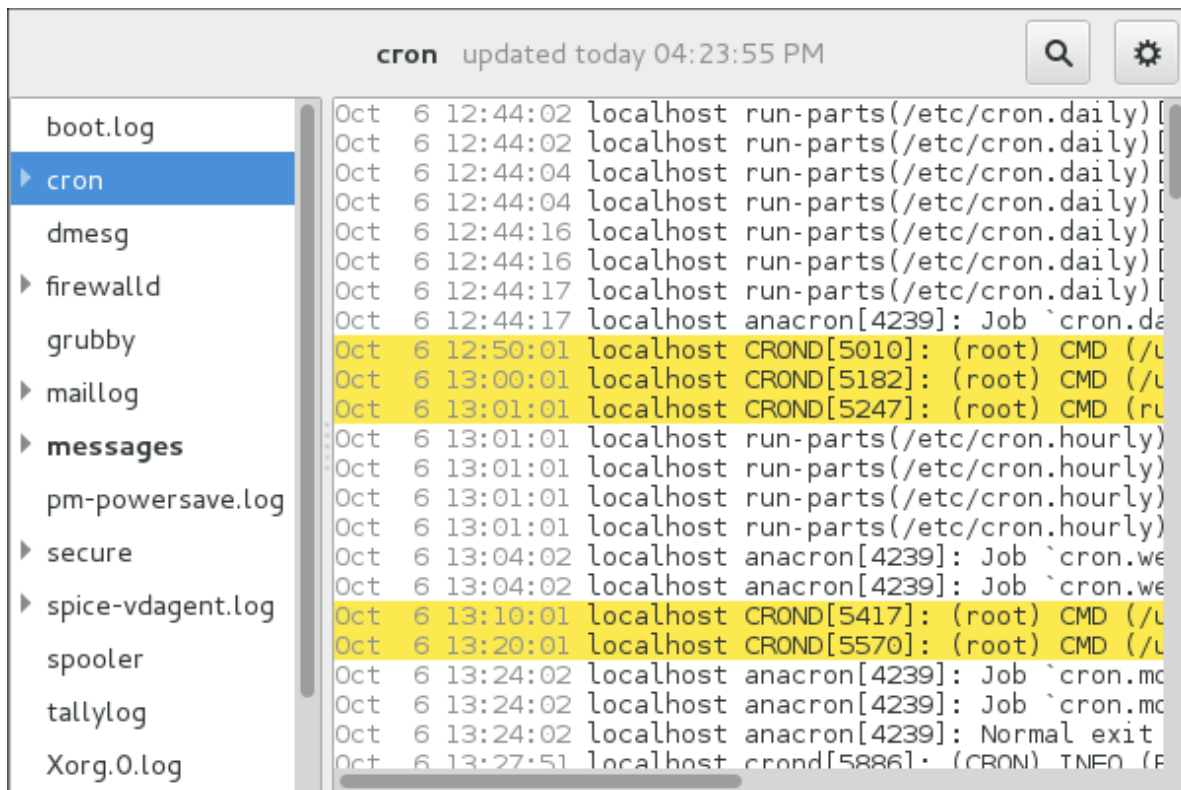


Figure 18.5. System Log - enabling a filter

When you select the **Show matches only** option, only the matched strings will be shown in the log file you are currently viewing.

18.12.2. 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 18.6](#), “[System Log - adding a log file](#)” illustrates the **Open Log** window.

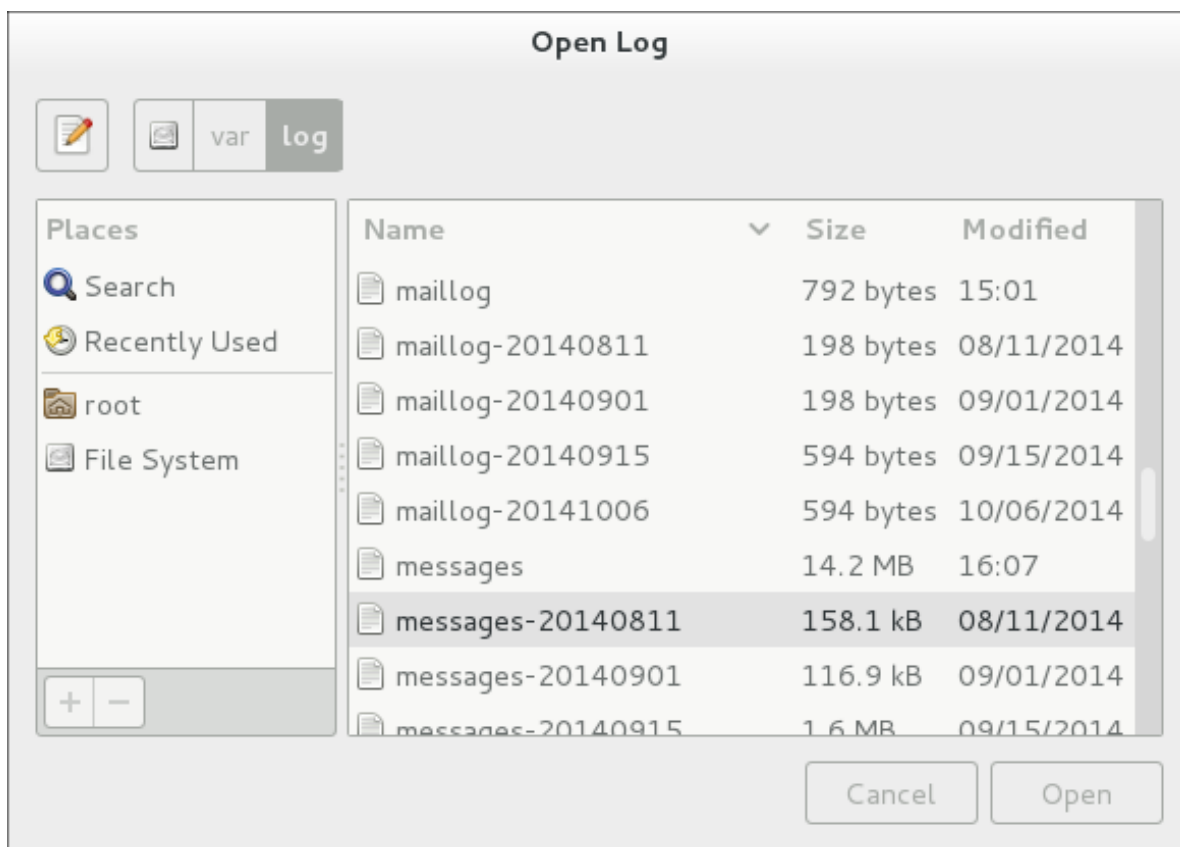


Figure 18.6. System Log - adding a log file

Click on the **Open** button to open the file. The file is immediately added to the viewing list where you can select it and view its contents.

Reading zipped log files

The **System Log** also allows you to open log files zipped in the **.gz** format.

18.12.3. Monitoring Log Files

System Log monitors all opened logs by default. If a new line is added to a monitored log file, the log name appears in bold in the log list. If the log file is selected or displayed, the new lines appear in bold at the bottom of the log file. [Figure 18.7, “System Log - new log alert”](#) illustrates a new alert in the **cron** log file and in the **messages** log file. Clicking on the **messages** log file displays the logs in the file with the new lines in bold.

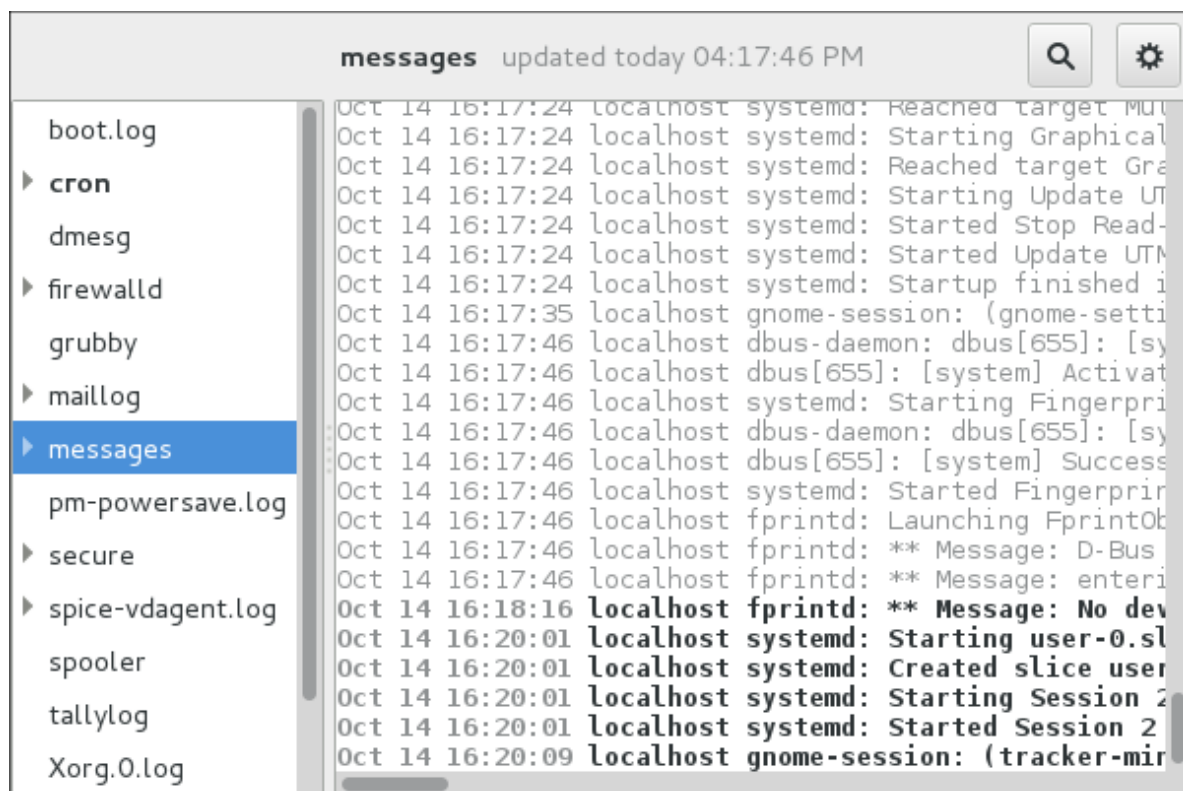


Figure 18.7. System Log - new log alert

18.13. Additional Resources

For more information on how to configure the `rsyslog` daemon and how to locate, view, and monitor log files, see the resources listed below.

Installed Documentation

- `rsyslogd(8)` — The manual page for the `rsyslogd` daemon documents its usage.
- `rsyslog.conf(5)` — The manual page named `rsyslog.conf` documents available configuration options.
- `logrotate(8)` — The manual page for the **logrotate** utility explains in greater detail how to configure and use it.
- `journalctl(1)` — The manual page for the **journalctl** daemon documents its usage.
- `journal.conf(5)` — This manual page documents available configuration options.
- `systemd.journal-fields(7)` — This manual page lists special **Journal** fields.

Online Documentation

- [rsyslog Home Page](http://www.rsyslog.com/)⁴ — The **rsyslog** home page offers a thorough technical breakdown of its features, documentation, configuration examples, and video tutorials.

⁴ <http://www.rsyslog.com/>

- [RainerScript documentation on the rsyslog Home Page](#)⁵ — Commented summary of data types, expressions, and functions available in *RainerScript*.
- [Description of queues on the rsyslog Home Page](#)⁶ — General information on various types of message queues and their usage.
- [rsyslog Wiki](#)⁷ — *The rsyslog Wiki* contains useful configuration examples.

⁵ <http://www.rsyslog.com/doc/rainerscript.html>

⁶ <http://www.rsyslog.com/doc/queues.html>

⁷ http://wiki.rsyslog.com/index.php/Main_Page

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.

Fedora 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.

Fedora comes with the following automated task utilities: **cron**, **anacron**, **at**, and **batch**.

Every utility is intended for scheduling a different job type: while Cron and Anacron schedule recurring jobs, At and Batch schedule one-time jobs (refer to [Section 19.1, “Cron and Anacron”](#) and [Section 19.2, “At and Batch”](#) respectively).

Fedora supports the use of `systemd.timer` for executing a job at a specific time. See `man systemd.timer(5)` for more information.

19.1. Cron and Anacron

Both Cron and Anacron are daemons that can schedule execution of recurring tasks to a certain point in time defined by the exact time, day of the month, month, day of the week, and week.

Cron jobs can run as often as every minute. However, the utility assumes that the system is running continuously and if the system is not on at the time when a job is scheduled, the job is not executed.

On the other hand, Anacron remembers the scheduled jobs if the system is not running at the time when the job is scheduled. The job is then executed as soon as the system is up. However, Anacron can only run a job once a day.

19.1.1. Installing Cron and Anacron

To install Cron and Anacron, you need to install the *cronie* package with Cron and the *cronie-anacron* package with Anacron (*cronie-anacron* is a sub-package of *cronie*).

To determine if the packages are already installed on your system, issue the following command:

```
rpm -q cronie cronie-anacron
```

The command returns full names of the *cronie* and *cronie-anacron* packages if already installed, or notifies you that the packages are not available.

To install these packages, use the **dnf** command in the following form as root:

```
dnf install package
```

For example, to install both Cron and Anacron, type the following at a shell prompt:

```
~]# dnf install cronie cronie-anacron
```

For more information on how to install new packages in Fedora, see [Section 6.2.4, “Installing Packages”](#).

19.1.2. Running the Crond Service

The cron and anacron jobs are both picked by the crond service. This section provides information on how to start, stop, and restart the crond service, and shows how to configure it to start automatically at boot time.

19.1.2.1. Starting and Stopping the Cron Service

To determine if the service is running, use the following command:

```
systemctl status crond.service
```

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

```
systemctl start crond.service
```

To configure the service to start automatically at boot time, use the following command as root:

```
systemctl enable crond.service
```

19.1.2.2. Stopping the Cron Service

To stop the crond service in the current session, type the following at a shell prompt as root:

```
systemctl stop crond.service
```

To prevent the service from starting automatically at boot time, use the following command as root:

```
systemctl disable crond.service
```

19.1.2.3. Restarting the Cron Service

To restart the crond service, type the following at a shell prompt as root:

```
systemctl restart crond.service
```

This command stops the service and starts it again in quick succession.

19.1.3. Configuring Anacron Jobs

The main configuration file to schedule jobs is the **/etc/anacrontab** file, which can be only accessed by the root user. The file contains the following:

```
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=45
# the jobs will be started during the following hours only
START_HOURS_RANGE=3-22

#period in days   delay in minutes   job-identifier   command
1                5                cron.daily      nice run-parts /etc/cron.daily
7                25               cron.weekly     nice run-parts /etc/cron.weekly
@monthly        45               cron.monthly    nice run-parts /etc/cron.monthly
```


The first three lines define the variables that configure the environment in which the anacron tasks run:

- **SHELL** — shell environment used for running jobs (in the example, the Bash shell)
- **PATH** — paths to executable programs
- **MAILTO** — username of the user who receives the output of the anacron jobs by email

If the **MAILTO** variable is not defined (**MAILTO=**), the email is not sent.

The next two variables modify the scheduled time for the defined jobs:

- **RANDOM_DELAY** — maximum number of minutes that will be added to the `delay in minutes` variable which is specified for each job

The minimum delay value is set, by default, to 6 minutes.

If **RANDOM_DELAY** is, for example, set to **12**, then between 6 and 12 minutes are added to the `delay in minutes` for each job in that particular anacrontab. **RANDOM_DELAY** can also be set to a value below **6**, including **0**. When set to **0**, no random delay is added. This proves to be useful when, for example, more computers that share one network connection need to download the same data every day.

- **START_HOURS_RANGE** — interval, when scheduled jobs can be run, in hours

In case the time interval is missed, for example due to a power failure, the scheduled jobs are not executed that day.

The remaining lines in the **/etc/anacrontab** file represent scheduled jobs and follow this format:

```
period in days    delay in minutes  job-identifier    command
```

- `period in days` — frequency of job execution in days

The property value can be defined as an integer or a macro (**@daily**, **@weekly**, **@monthly**), where **@daily** denotes the same value as integer 1, **@weekly** the same as 7, and **@monthly** specifies that the job is run once a month regardless of the length of the month.

- `delay in minutes` — number of minutes anacron waits before executing the job

The property value is defined as an integer. If the value is set to **0**, no delay applies.

- `job-identifier` — unique name referring to a particular job used in the log files
- `command` — command to be executed

The command can be either a command such as **ls /proc >> /tmp/proc** or a command which executes a custom script.

Any lines that begin with a hash sign (#) are comments and are not processed.

19.1.3.1. Examples of Anacron Jobs

The following example shows a simple **/etc/anacrontab** file:

```
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
```



```
# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=30
# the jobs will be started during the following hours only
START_HOURS_RANGE=16-20

#period in days   delay in minutes   job-identifier   command
1                20      dailyjob       nice run-parts /etc/cron.daily
7                25      weeklyjob       /etc/weeklyjob.bash
@monthly        45      monthlyjob      ls /proc >> /tmp/proc
```

All jobs defined in this **anacrontab** file are randomly delayed by 6-30 minutes and can be executed between 16:00 and 20:00.

The first defined job is triggered daily between 16:26 and 16:50 (**RANDOM_DELAY** is between 6 and 30 minutes; the delay in minutes property adds 20 minutes). The command specified for this job executes all present programs in the **/etc/cron.daily/** directory using the **run-parts** script (the **run-parts** scripts accepts a directory as a command-line argument and sequentially executes every program in the directory). See the **run-parts** man page for more information on the **run-parts** script.

The second job executes the **weeklyjob.bash** script in the **/etc/** directory once a week.

The third job runs a command, which writes the contents of **/proc** to the **/tmp/proc** file (**ls /proc >> /tmp/proc**) once a month.

19.1.4. Configuring Cron Jobs

The configuration file for cron jobs is **/etc/crontab**, which can be only modified by the root user. The file contains the following:

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/
# For details see man 4 crontabs
# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .----- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * * user-name command to be executed
```

The first three lines contain the same variable definitions as an **anacrontab** file: **SHELL**, **PATH**, and **MAILTO**. For more information about these variables, see [Section 19.1.3, “Configuring Anacron Jobs”](#).

In addition, the file can define the **HOME** variable. The **HOME** variable defines the directory, which will be used as the home directory when executing commands or scripts run by the job.

The remaining lines in the **/etc/crontab** file represent scheduled jobs and have the following format:

minute	hour	day	month	day of week	username	command
--------	------	-----	-------	-------------	----------	---------

The following define the time when the job is to be run:

- minute — any integer from 0 to 59

- **hour** — any integer from 0 to 23
- **day** — any integer from 1 to 31 (must be a valid day if a month is specified)
- **month** — any integer from 1 to 12 (or the short name of the month such as jan or feb)
- **day of week** — any integer from 0 to 7, where 0 or 7 represents Sunday (or the short name of the week such as sun or mon)

The following define other job properties:

- **username** — specifies the user under which the jobs are run.
- **command** — the command to be executed.

The command can be either a command such as **ls /proc /tmp/proc** or a command which executes a custom script.

For any of the above values, an asterisk (*) can be used to specify all valid values. If you, for example, define the month value as an asterisk, the job will be executed every month within the constraints of the other values.

A hyphen (-) between integers specifies a range of integers. For example, **1-4** means the integers 1, 2, 3, and 4.

A list of values separated by commas (,) specifies a list. For example, **3,4,6,8** indicates exactly these four integers.

The forward slash (/) can be used to specify step values. The value of an integer will be skipped within a range following the range with **/integer**. For example, the minute value defined as **0-59/2** denotes every other minute in the minute field. Step values can also be used with an asterisk. For instance, if the month value is defined as ***/3**, the task will run every third month.

Any lines that begin with a hash sign (#) are comments and are not processed.

Users other than **root** can configure cron tasks with the **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 specific 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 the user name and written to the file **/var/spool/cron/username**. To list the contents of the current user's crontab file, use the **crontab -l** command.

The **/etc/cron.d/** directory contains files that have the same syntax as the **/etc/crontab** file. Only **root** is allowed to create and modify files in this directory.



Do not restart the daemon to apply the changes

The cron daemon checks the **/etc/anacrontab** file, the **/etc/crontab** file, the **/etc/cron.d/** directory, and the **/var/spool/cron/** directory every minute for changes and the detected changes are loaded into memory. It is therefore not necessary to restart the daemon after an anacrontab or a crontab file have been changed.

19.1.5. Controlling Access to Cron

To restrict the access to Cron, you can use the `/etc/cron.allow` and `/etc/cron.deny` files. These access control files use the same format with one user name on each line. Mind that no whitespace characters are permitted in either file.

If the `cron.allow` file exists, only users listed in the file are allowed to use cron, and the `cron.deny` file is ignored.

If the `cron.allow` file does not exist, users listed in the `cron.deny` file are not allowed to use Cron.

The Cron daemon (`crond`) does not have to be restarted if the access control files are modified. The access control files are checked each time a user tries to add or delete a cron job.

The root user can always use cron, regardless of the user names listed in the access control files.

You can control the access also through Pluggable Authentication Modules (PAM). The settings are stored in the `/etc/security/access.conf` file. For example, after adding the following line to the file, no other user but the root user can create crontabs:

```
-:ALL EXCEPT root :cron
```

The forbidden jobs are logged in an appropriate log file or, when using `crontab -e`, returned to the standard output. For more information, see the `access.conf.5` manual page.

19.1.6. Black and White Listing of Cron Jobs

Black and white listing of jobs is used to define parts of a job that do not need to be executed. This is useful when calling the `run-parts` script on a Cron directory, such as `/etc/cron.daily/`: if the user adds programs located in the directory to the job black list, the `run-parts` script will not execute these programs.

To define a black list, create a `jobs.deny` file in the directory that `run-parts` scripts will be executing from. For example, if you need to omit a particular program from `/etc/cron.daily/`, create the `/etc/cron.daily/jobs.deny` file. In this file, specify the names of the programs to be omitted from execution (only programs located in the same directory can be enlisted). If a job runs a command which runs the programs from the `/etc/cron.daily/` directory, such as `run-parts /etc/cron.daily`, the programs defined in the `jobs.deny` file will not be executed.

To define a white list, create a `jobs.allow` file.

The principles of `jobs.deny` and `jobs.allow` are the same as those of `cron.deny` and `cron.allow` described in section [Section 19.1.5, “Controlling Access to Cron”](#).

19.2. At and Batch

While Cron is used to schedule recurring tasks, the **At** utility is used to schedule a one-time task at a specific time and the **Batch** utility is used to schedule a one-time task to be executed when the system load average drops below 0.8.

19.2.1. Installing At and Batch

To determine if the `at` package is already installed on your system, issue the following command:

```
rpm -q at
```


The command returns the full name of the *at* package if already installed or notifies you that the package is not available.

To install the packages, use the **dnf** command in the following form as root:

```
dnf install package
```

For example, to install both At and Batch, type the following at a shell prompt:

```
~]# dnf install at
```

For more information on how to install new packages in Fedora, see [Section 6.2.4, “Installing Packages”](#).

19.2.2. Running the At Service

The At and Batch jobs are both picked by the *atd* service. This section provides information on how to start, stop, and restart the *atd* service, and shows how to configure it to start automatically at boot time.

19.2.2.1. Starting and Stopping the At Service

To determine if the service is running, use the following command:


```
systemctl status atd.service
```

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

```
systemctl start atd.service
```

To configure the service to start automatically at boot time, use the following command as root:

```
systemctl enable atd.service
```

 **Note**

It is recommended that you configure your system to start the *atd* service automatically at boot time.

19.2.2.2. Stopping the At Service

To stop the *atd* service, type the following at a shell prompt as root:

```
systemctl stop atd.service
```

To prevent the service from starting automatically at boot time, use the following command as root:

```
systemctl disable atd.service
```


19.2.2.3. Restarting the At Service

To restart the `atd` service, type the following at a shell prompt as root:

```
systemctl restart atd.service
```

This command stops the service and starts it again in quick succession.

19.2.3. Configuring an At Job

To schedule a one-time job for a specific time with the **At** utility, do the following:

1. On the command line, type the command **at** *TIME*, where *TIME* is the time when the command is to be executed.

The *TIME* argument can be defined in any of the following formats:

- **HH:MM** specifies the exact hour and minute; For example, **04:00** specifies 4:00 a.m.
- **midnight** specifies 12:00 a.m.
- **noon** specifies 12:00 p.m.
- **teatime** specifies 4:00 p.m.
- **MONTHDAYYEAR** format; For example, **January 15 2012** specifies the 15th day of January in the year 2012. The year value is optional.
- **MMDDYY**, **MM/DD/YY**, or **MM.DD.YY** formats; For example, **011512** for the 15th day of January in the year 2012.
- **now + TIME** where *TIME* is defined as an integer and the value type: minutes, hours, days, or weeks. For example, **now + 5 days** specifies that the command will be executed at the same time five days from now.

The time must be specified first, followed by the optional date. For more information about the time format, see the `/usr/share/doc/at-<version>/timespec` text file.

If the specified time has past, the job is executed at the time the next day.

2. In the displayed `at>` prompt, define the job commands:

Type the command the job should execute and press **Enter**. Optionally, repeat the step to provide multiple commands.

Enter a shell script at the prompt and press **Enter** after each line in the script.

The job will use the shell set in the user's SHELL environment, the user's login shell, or `/bin/sh` (whichever is found first).

3. Once finished, press **Ctrl+D** on an empty line to exit the prompt.

If the set of commands or the script tries to display information to standard output, the output is emailed to the user.

To view the list of pending jobs, use the **atq** command. See [Section 19.2.5, “Viewing Pending Jobs”](#) for more information.

You can also restrict the usage of the **at** command. For more information, see [Section 19.2.7, “Controlling Access to At and Batch”](#) for details.

19.2.4. Configuring a Batch Job

The **Batch** application executes the defined one-time tasks when the system load average decreases below 0.8.

To define a Batch job, do the following:

1. On the command line, type the command **batch**.
2. In the displayed **at>** prompt, define the job commands:

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 19.2.5, “Viewing Pending Jobs”](#) for more information.

You can also restrict the usage of the **batch** command. For more information, see [Section 19.2.7, “Controlling Access to At and Batch”](#) for details.

19.2.5. Viewing Pending Jobs

To view the pending **At** and **Batch** jobs, run the **atq** command. The **atq** command displays a list of pending jobs, with each job on a separate line. Each line follows the job number, date, hour, job class, and user name format. Users can only view their own jobs. If the **root** user executes the **atq** command, all jobs for all users are displayed.

19.2.6. Additional Command Line Options

Additional command line options for **at** and **batch** include the following:

Table 19.1. **at** and **batch** Command Line Options

Option	Description
-f	Read the commands or shell script from a file instead of specifying them at the prompt.
-m	Send email to the user when the job has been completed.
-v	Display the time that the job is executed.

19.2.7. Controlling Access to At and Batch

You can restrict the access to the **at** and **batch** commands using the **/etc/at.allow** and **/etc/at.deny** files. These access control files use the same format defining one user name on each line. Mind that no whitespace are permitted in either file.

If the file **at.allow** exists, only users listed in the file are allowed to use **at** or **batch**, and the **at.deny** file is ignored.

If **at.allow** does not exist, users listed in **at.deny** are not allowed to use **at** or **batch**.

The **at** daemon (**atd**) does not have to be restarted if the access control files are modified. The access control files are read each time a user tries to execute the **at** or **batch** commands.

The root user can always execute **at** and **batch** commands, regardless of the content of the access control files.

19.3. Additional Resources

To learn more about configuring automated tasks, see the following installed documentation:

- **cron(8)** man page contains an overview of cron.
- **crontab** man pages in sections 1 and 5:
 - The manual page in section 1 contains an overview of the **crontab** file.
 - The man page in section 5 contains the format for the file and some example entries.
- **anacron(8)** manual page contains an overview of anacron.
- **anacrontab(5)** manual page contains an overview of the **anacrontab** file.
- **run-parts(4)** manual page contains an overview of the **run-parts** script.
- **/usr/share/doc/at/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.

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 Fedora system, the **oprofile** package must be installed to use this tool.

Many processors include dedicated performance monitoring hardware. This hardware makes it possible to detect when certain events happen (such as the requested data not being in cache). The hardware normally takes the form of one or more *counters* that are incremented each time an event takes place. When the counter value increments, an interrupt is generated, making it possible to control the amount of detail (and therefore, overhead) produced by performance monitoring.

OProfile uses this hardware (or a timer-based substitute in cases where performance monitoring hardware is not present) to collect *samples* of performance-related data each time a counter generates an interrupt. These samples are periodically written out to disk; later, the data contained in these samples can then be used to generate reports on system-level and application-level performance.

Be aware of the following limitations when using OProfile:

- *Use of shared libraries* — Samples for code in shared libraries are not attributed to the particular application unless the **--separate=library** option is used.
- *Performance monitoring samples are inexact* — When a performance monitoring register triggers a sample, the interrupt handling is not precise like a divide by zero exception. Due to the out-of-order execution of instructions by the processor, the sample may be recorded on a nearby instruction.
- *opreport does not associate samples for inline functions properly* — **opreport** uses a simple address range mechanism to determine which function an address is in. Inline function samples are not attributed to the inline function but rather to the function the inline function was inserted into.
- *OProfile accumulates data from multiple runs* — OProfile is a system-wide profiler and expects processes to start up and shut down multiple times. Thus, samples from multiple runs accumulate. Use the command **opcontrol --reset** to clear out the samples from previous runs.
- *Hardware performance counters do not work on guest virtual machines* — Because the hardware performance counters are not available on virtual systems, you need to use the **timer** mode. Enter 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).

20.1. Overview of Tools

[Table 20.1, “OProfile Commands”](#) provides a brief overview of the most commonly used tools provided with the **oprofile** package.

Table 20.1. OProfile Commands

Command	Description
ophelp	Displays available events for the system's processor along with a brief description of each.
opimport	Converts sample database files from a foreign binary format to the native format for the system. Only use this option when analyzing a sample database from a different architecture.
opannotate	Creates annotated source for an executable if the application was compiled with debugging symbols. See Section 20.6.4, “Using opannotate” for details.
opcontrol	Configures what data is collected. See Section 20.3, “Configuring OProfile Using Legacy Mode” for details.
operf	Recommended tool to be used in place of opcontrol for profiling. See Section 20.2, “Using operf” for details. For differences between operf and opcontrol see Section 20.1.1, “operf vs. opcontrol” .
opreport	Retrieves profile data. See Section 20.6.1, “Using opreport” for details.
oprofiled	Runs as a daemon to periodically write sample data to disk.

20.1.1. operf vs. opcontrol

There are two mutually exclusive methods for collecting profiling data with OProfile. You can either use the newer and preferred **operf** or the **opcontrol** tool.

operf

This is the recommended mode for profiling. The **operf** tool uses the Linux Performance Events Subsystem, and therefore does not require the *oprofile* kernel driver. The **operf** tool allows you to target your profiling more precisely, as a single process or system-wide, and also allows OProfile to co-exist better with other tools using the performance monitoring hardware on your system. Unlike **opcontrol**, it can be used without the root privileges. However, **operf** is also capable of system-wide operations with use of the **--system-wide** option, where root authority is required.

With **operf**, there is no initial setup needed. You can invoke **operf** with command-line options to specify your profiling settings. After that, you can run the OProfile post-processing tools described in [Section 20.6, “Analyzing the Data”](#). See [Section 20.2, “Using operf”](#) for further information.

opcontrol

This mode consists of the **opcontrol** shell script, the *oprofiled* daemon, and several post-processing tools. The **opcontrol** command is used for configuring, starting, and stopping a profiling session. An OProfile kernel driver, usually built as a kernel module, is used for collecting samples, which are then recorded into sample files by *oprofiled*. You can use legacy mode only if you have root privileges. In certain cases, such as when you need to sample areas with disabled interrupt request (IRQ), this is a better alternative.

Before OProfile can be run in legacy mode, it must be configured as shown in [Section 20.3, “Configuring OProfile Using Legacy Mode”](#). These settings are then applied when starting OProfile ([Section 20.4, “Starting and Stopping OProfile Using Legacy Mode”](#)).

20.2. Using `operf`

operf is the recommended profiling mode that does not require initial setup before starting. All settings are specified as command-line options and there is no separate command to start the profiling process. To stop **operf**, press Ctrl+C. The typical **operf** command syntax looks as follows:

```
operf options range command args
```

Replace *options* with the desired command-line options to specify your profiling settings. Full set of options is described in `operf(1)` manual page. Replace *range* with one of the following:

--system-wide - this setting allows for global profiling, see [Using `operf` in System-wide Mode](#)

--pid=PID - this is to profile a running application, where *PID* is the process ID of the process you want to profile.

With *command* and *args*, you can define a specific command or application to be profiled, and also the input arguments that this command or application requires. Either *command*, **--pid** or **--system-wide** is required, but these cannot be used simultaneously.

When you invoke **operf** on a command line without setting the *range* option, data will be collected for the children processes.

Using `operf` in System-wide Mode

To run **operf --system-wide**, you need root authority. When finished profiling, you can stop **operf** with Ctrl+C.

If you run **operf --system-wide** as a background process (with `&`), stop it in a controlled manner in order to process the collected profile data. For this purpose, use:

```
kill -SIGINT operf-PID
```

When running **operf --system-wide**, it is recommended that your current working directory is `/root` or a subdirectory of `/root` so that sample data files are not stored in locations accessible by regular users.

20.2.1. Specifying the Kernel

To monitor the kernel, execute the following command:

```
operf --vmlinux=vmlinux_path
```

With this option, you can specify a path to a `vmlinux` file that matches the running kernel. Kernel samples will be attributed to this binary, allowing post-processing tools to attribute samples to the appropriate kernel symbols. If this option is not specified, all kernel samples will be attributed to a pseudo binary named "no-vmlinux".

20.2.2. Setting Events to Monitor

Most processors contain counters, which are used by OProfile to monitor specific events. As shown in [Table 20.3, "OProfile Processors and Counters"](#), the number of counters available depends on the processor.

The events for each counter can be configured via the command line or with a graphical interface. For more information on the graphical interface, see [Section 20.10, “Graphical Interface”](#). If the counter cannot be set to a specific event, an error message is displayed.

Older Processors and **operf**

Some older processor models are not supported by the underlying Linux Performance Events Subsystem kernel and therefore are not supported by **operf**. If you receive this message:

```
Your kernel's Performance Events Subsystem does not support your processor type
```

when attempting to use **operf**, try profiling with **opcontrol** to see if your processor type may be supported by OProfile's legacy mode.

Using **operf** on Virtual Systems

Since hardware performance counters are not available on guest virtual machines, you have to enable *timer* mode to use **operf** on virtual systems. To do so, type as root:

```
opcontrol --deinit
```

```
modprobe oprofile timer=1
```

To set the event for each configurable counter via the command line, use:

```
operf --events=event1,event2...
```

Here, pass a comma-separated list of event specifications for profiling. Each event specification is a colon-separated list of attributes in the following form:

```
event-name:sample-rate:unit-mask:kernel:user
```

[Table 20.2, “Event Specifications”](#) summarizes these options. The last three values are optional, if you omit them, they will be set to their default values. Note that certain events do require a unit mask.

Table 20.2. Event Specifications

Specification	Description
<i>event-name</i>	The exact symbolic event name taken from ophelp
<i>sample-rate</i>	The number of events to wait before sampling again. The smaller the count, the more frequent the samples. For events that do not happen frequently, a lower count may be needed to capture a statistically significant number of event instances. On the other hand, sampling too

Specification	Description
	frequently can overload the system. By default, OProfile uses a time-based event set, which creates a sample every 100,000 clock cycles per processor.
<i>unit-mask</i>	Unit masks, which further define the event, are listed in ophelp . You can insert either a hexadecimal value, beginning with "0x", or a string that matches the first word of the unit mask description in ophelp . Definition by name is valid only for unit masks having "extra:" parameters, as shown by the output of ophelp . This type of unit mask cannot be defined with a hexadecimal value. Note that on certain architectures, there can be multiple unit masks with the same hexadecimal value. In that case they have to be specified by their names only.
<i>kernel</i>	Specifies whether to profile kernel code (insert 0 or 1 (default))
<i>user</i>	Specifies whether to profile user-space code (insert 0 or 1 (default))

The events available vary depending on the processor type. When no event specification is given, the default event for the running processor type will be used for profiling. See [Table 20.4, "Default Events"](#) for a list of these default events. To determine the events available for profiling, use the **ophelp** command.

```
ophelp
```

20.2.3. Categorization of Samples

The **--separate-thread** option categorizes samples by thread group ID (tgid) and thread ID (tid). This is useful for seeing per-thread samples in multi-threaded applications. When used in conjunction with the **--system-wide** option, **--separate-thread** is also useful for seeing per-process (i.e., per-thread group) samples for the case where multiple processes are executing the same program during a profiling run.

The **--separate-cpu** option categorizes samples by CPU.

20.3. Configuring OProfile Using Legacy Mode

Before OProfile can be run in legacy mode, it must be configured. At a minimum, selecting to monitor the kernel (or selecting not to monitor the kernel) is required. The following sections describe how to use the **opcontrol** utility to configure OProfile. As the **opcontrol** commands are executed, the setup options are saved to the `/root/.oprofile/daemonrc` file.

20.3.1. Specifying the Kernel

First, configure whether OProfile should monitor the kernel. This is the only configuration option that is required before starting OProfile. All others are optional.

To monitor the kernel, execute the following command as root:

```
~]# opcontrol --setup --vmlinux=/usr/lib/debug/lib/modules/`uname -r`/vmlinux
```



Install the debuginfo package

In order to monitor the kernel, the *debuginfo* package which contains the uncompressed kernel must be installed.

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 20.7, “Understanding the /dev/oprofile/ directory”](#) 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 20.3.3, “Separating Kernel and User-space Profiles”](#).

20.3.2. Setting Events to Monitor

Most processors contain *counters*, which are used by OProfile to monitor specific events. As shown in [Table 20.3, “OProfile Processors and Counters”](#), the number of counters available depends on the processor.

Table 20.3. OProfile Processors and Counters

Processor	cpu_type	Number of Counters
AMD64	x86-64/hammer	4
AMD Family 10h	x86-64/family10	4
AMD Family 11h	x86-64/family11	4
AMD Family 12h	x86-64/family12	4
AMD Family 14h	x86-64/family14	4
AMD Family 15h	x86-64/family15	6
Applied Micro X-Gene	arm/armv8-xgene	4
ARM Cortex A53	arm/armv8-ca53	6
ARM Cortex A57	arm/armv8-ca57	6
IBM eServer System i and IBM eServer System p	timer	1
IBM POWER4	ppc64/power4	8
IBM POWER5	ppc64/power5	6

Processor	cpu_type	Number of Counters
IBM PowerPC 970	ppc64/970	8
IBM PowerPC 970MP	ppc64/970MP	8
IBM POWER5+	ppc64/power5+	6
IBM POWER5++	ppc64/power5++	6
IBM POWER56	ppc64/power6	6
IBM POWER7	ppc64/power7	6
IBM POWER8	ppc64/power7	8
IBM S/390 and IBM System z	timer	1
Intel Core i7	i386/core_i7	4
Intel Nehalem microarchitecture	i386/nehalem	4
Intel Westmere microarchitecture	i386/westmere	4
Intel Haswell microarchitecture (non-hyper-threaded)	i386/haswell	8
Intel Haswell microarchitecture (hyper-threaded)	i386/haswell-ht	4
Intel Ivy Bridge microarchitecture (non-hyper-threaded)	i386/ivybridge	8
Intel Ivy Bridge microarchitecture (hyper-threaded)	i386/ivybridge-ht	4
Intel Sandy Bridge microarchitecture (non-hyper-threaded)	i386/sandybridge	8
Intel Sandy Bridge microarchitecture	i386/sandybridge-ht	4
Intel Broadwell microarchitecture (non-hyper-threaded)	i386/broadwell	8
Intel Broadwell microarchitecture (hyper-threaded)	i386/broadwell-ht	4
Intel Silvermont microarchitecture	i386/silvermont	2
TIMER_INT	timer	1

Use [Table 20.3, “OProfile Processors and Counters”](#) to determine the number of events that can be monitored simultaneously for your CPU type. If the processor does not have supported performance monitoring hardware, the **timer** is used as the processor type.

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 0 are not set to an event by default. The default events monitored are shown in [Table 20.4, “Default Events”](#).

Table 20.4. Default Events

Processor	Default Event for Counter	Description
AMD Athlon and AMD64	CPU_CLK_UNHALTED	The processor's clock is not halted

Processor	Default Event for Counter	Description
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
Applied Micro X-Gene	CPU_CYCLES	Processor Cycles
ARM Cortex A53	CPU_CYCLES	Processor Cycles
ARM Cortex A57	CPU_CYCLES	Processor Cycles
IBM POWER4	CYCLES	Processor Cycles
IBM POWER5	CYCLES	Processor Cycles
IBM POWER8	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
Intel Broadwell microarchitecture	CPU_CLK_UNHALTED	The processor's clock is not halted
Intel Silvermont microarchitecture	CPU_CLK_UNHALTED	The processor's clock is not halted
TIMER_INT	(none)	Sample for each timer interrupt

The number of events that can be monitored at one time is determined by the number of counters for the processor. However, it is not a one-to-one correlation; on some processors, certain events must be mapped to specific counters. To determine the number of counters available, execute the following command:

```
~]# ls -d /dev/oprofile/[0-9]*
```

The events available vary depending on the processor type. To determine the events available for profiling, execute the following command as root (the list is specific to the system's processor type):

```
~]# ophelp
```


Make sure that OProfile is configured

Unless OProfile is properly configured, **ophe1p** 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 20.3, “Configuring OProfile Using Legacy Mode”](#).

The events for each counter can be configured via the command line or with a graphical interface. For more information on the graphical interface, see [Section 20.10, “Graphical Interface”](#). If the counter cannot be set to a specific event, an error message is displayed.

To set the event for each configurable counter via the command line, use **opcontrol**:

```
~]# opcontrol --event=event-name:sample-rate
```

Replace *event-name* with the exact name of the event from **ophe1p**, and replace *sample-rate* with the number of events between samples.

20.3.2.1. Sampling Rate

By default, a time-based event set is selected. It creates a sample every 100,000 clock cycles per processor. If the timer interrupt is used, the timer is set to the respective rate and is not user-settable. If the **cpu_type** is not **timer**, each event can have a *sampling rate* set for it. The sampling rate is the number of events between each sample snapshot.

When setting the event for the counter, a sample rate can also be specified:

```
~]# opcontrol --event=event-name:sample-rate
```

Replace *sample-rate* with the number of events to wait before sampling again. The smaller the count, the more frequent the samples. For events that do not happen frequently, a lower count may be needed to capture the event instances.



Sampling too frequently can overload the system

Be extremely careful when setting sampling rates. Sampling too frequently can overload the system, causing the system to appear frozen or causing the system to actually freeze.

20.3.2.2. Unit Masks

Some user performance monitoring events may also require unit masks to further define the event.

Unit masks for each event are listed with the **ophelp** command. The values for each unit mask are listed in hexadecimal format. To specify more than one unit mask, the hexadecimal values must be combined using a bitwise *or* operation.

```
~]# opcontrol --event=event-name:sample-rate:unit-mask
```

Note that on certain architectures, there can be multiple unit masks with the same hexadecimal value. In that case they have to be specified by their names only.

20.3.3. Separating Kernel and User-space Profiles

By default, kernel mode and user mode information is gathered for each event. To configure OProfile to ignore events in kernel mode for a specific counter, execute the following command:

```
~]# opcontrol --event=event-name:sample-rate:unit-mask:0
```

Execute the following command to start profiling kernel mode for the counter again:

```
~]# opcontrol --event=event-name:sample-rate:unit-mask:1
```

To configure OProfile to ignore events in user mode for a specific counter, execute the following command:

```
~]# opcontrol --event=event-name:sample-rate:unit-mask:1:0
```

Execute the following command to start profiling user mode for the counter again:

```
~]# opcontrol --event=event-name:sample-rate:unit-mask:1:1
```

When the OProfile daemon writes the profile data to sample files, it can separate the kernel and library profile data into separate sample files. To configure how the daemon writes to sample files, execute the following command as root:

```
~]# opcontrol --separate=choice
```

The *choice* argument can be one of the following:

- **none** — Do not separate the profiles (default).
- **library** — Generate per-application profiles for libraries.
- **kernel** — Generate per-application profiles for the kernel and kernel modules.
- **all** — Generate per-application profiles for libraries and per-application profiles for the kernel and kernel modules.

If **--separate=library** is used, the sample file name includes the name of the executable as well as the name of the library.



Restart the OProfile profiler

These configuration changes will take effect when the OProfile profiler is restarted.

20.4. Starting and Stopping OProfile Using Legacy Mode

To start monitoring the system with OProfile, execute the following command as root:

```
~]# opcontrol --start
```

Output similar to the following is displayed:

```
Using log file /var/lib/oprofile/oprofiled.log Daemon started. Profiler running.
```

The settings in `/root/.oprofile/daemonrc` are used.

The OProfile daemon, **oprofiled**, is started; it periodically writes the sample data to the `/var/lib/oprofile/samples/` directory. The log file for the daemon is located at `/var/lib/oprofile/oprofiled.log`.



Disable the nmi_watchdog registers

On a Fedora 26 system, the **nmi_watchdog** registers with the **perf** subsystem. Due to this, the **perf** subsystem grabs control of the performance counter registers at boot time, blocking OProfile from working.

To resolve this, either boot with the **nmi_watchdog=0** kernel parameter set, or run the following command as root to disable **nmi_watchdog** at run time:

```
~]# echo 0 > /proc/sys/kernel/nmi_watchdog
```

To re-enable **nmi_watchdog**, use the following command as root:

```
~]# echo 1 > /proc/sys/kernel/nmi_watchdog
```

To stop the profiler, execute the following command as root:

```
~]# opcontrol --shutdown
```

20.5. Saving Data in Legacy Mode

Sometimes it is useful to save samples at a specific time. For example, when profiling an executable, it may be useful to gather different samples based on different input data sets. If the number of events to be monitored exceeds the number of counters available for the processor, multiple runs of OProfile can be used to collect data, saving the sample data to different files each time.

To save the current set of sample files, execute the following command, replacing *name* with a unique descriptive name for the current session:

```
~]# opcontrol --save=name
```

The command creates the directory `/var/lib/oprofile/samples/name/` and the current sample files are copied to it.

To specify the session directory to hold the sample data, use the `--session-dir` option. If not specified, the data is saved in the `oprofile_data/` directory on the current path.

20.6. Analyzing the Data

The same OProfile post-processing tools are used whether you collect your profile with **operf** or **opcontrol** in legacy mode.

By default, **operf** stores the profiling data in the `current_dir/oprofile_data/` directory. You can change to a different location with the `--session-dir` option. The usual post-profiling analysis tools such as **opreport** and **opannotate** can be used to generate profile reports. These tools search for samples in `current_dir/oprofile_data/` first. If this directory does not exist, the analysis tools use the standard session directory of `/var/lib/oprofile/`. Statistics, such as total samples received and lost samples, are written to the `session_dir/samples/operf.log` file.

When using legacy mode, the OProfile daemon, **oprofiled**, periodically collects the samples and writes them to the `/var/lib/oprofile/samples/` directory. Before reading the data, make sure all data has been written to this directory by executing the following command as root:

```
~]# opcontrol --dump
```

Each sample file name is based on the name of the executable. For example, the samples for the default event on a Pentium III processor for `/bin/bash` becomes:

```
\{root\}/bin/bash/\{dep\}/\{root\}/bin/bash/CPU_CLK_UNHALTED.100000
```

The following tools are available to profile the sample data once it has been collected:

- **opreport**
- **opannotate**

Use these tools, along with the binaries profiled, to generate reports that can be further analyzed.



Back up the executable and the sample files

The executable being profiled must be used with these tools to analyze the data. If it must change after the data is collected, back up the executable used to create the samples as well as the sample files. Note that the names of the sample file and the binary have to agree. You cannot make a backup if these names do not match. As an alternative, **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 20.5, “Saving Data in Legacy Mode”](#) for details on how to back up the sample file.

20.6.1. Using **opreport**

The **opreport** tool provides an overview of all the executables being profiled. The following is part of a sample output from the **opreport** command:

```
~]$ opreport
Profiling through timer interrupt
TIMER:0|
samples|      %|
-----
25926 97.5212 no-vmlinux
359  1.3504 pi
65   0.2445 Xorg
62   0.2332 libvte.so.4.4.0
56   0.2106 libc-2.3.4.so
34   0.1279 libglib-2.0.so.0.400.7
19   0.0715 libXft.so.2.1.2
17   0.0639 bash
8    0.0301 ld-2.3.4.so
8    0.0301 libgdk-x11-2.0.so.0.400.13
6    0.0226 libgobject-2.0.so.0.400.7
5    0.0188 oprofiled
4    0.0150 libpthread-2.3.4.so
4    0.0150 libgtk-x11-2.0.so.0.400.13
3    0.0113 libXrender.so.1.2.2
3    0.0113 du
1    0.0038 libcrypto.so.0.9.7a
1    0.0038 libpam.so.0.77
1    0.0038 libtermcap.so.2.0.8
1    0.0038 libX11.so.6.2
1    0.0038 libgthread-2.0.so.0.400.7
1    0.0038 libwnck-1.so.4.9.0
```

Each executable is listed on its own line. The first column is the number of samples recorded for the executable. The second column is the percentage of samples relative to the total number of samples. The third column is the name of the executable.

See the `opreport(1)` manual page for a list of available command-line options, such as the `-r` option used to sort the output from the executable with the smallest number of samples to the one with the largest number of samples. You can also use the `-t` or `--threshold` option to trim the output of `opcontrol`.

20.6.2. Using `opreport` on a Single Executable

To retrieve more detailed profiled information about a specific executable, use the `opreport` command:

```
~]# opreport mode executable
```

Replace *executable* with the full path to the executable to be analyzed. *mode* stands for one of the following options:

-l

This option is used to list sample data by symbols. For example, running this command:

```
~]# opreport -l /lib/tls/libc-version.so
```

produces the following output:

```
samples % symbol name
12 21.4286 __conv_transform_utf8_internal
5 8.9286 _int_malloc 4 7.1429 malloc
3 5.3571 __i686.get_pc_thunk.bx
3 5.3571 _dl_mcount_wrapper_check
3 5.3571 mbrtowc
3 5.3571 memcpy
2 3.5714 _int_realloc
2 3.5714 _nl_intern_locale_data
2 3.5714 free
2 3.5714 strcmp
1 1.7857 __ctype_get_mb_cur_max
1 1.7857 __unregister_atfork
1 1.7857 __write_nocancel
1 1.7857 _dl_addr
1 1.7857 _int_free
1 1.7857 _itoa_word
1 1.7857 calc_eclosure_iter
1 1.7857 fopen@@GLIBC_2.1
1 1.7857 getpid
1 1.7857 memmove
1 1.7857 msort_with_tmp
1 1.7857 strcpy
1 1.7857 strlen
1 1.7857 vfprintf
1 1.7857 write
```

The first column is the number of samples for the symbol, the second column is the percentage of samples for this symbol relative to the overall samples for the executable, and the third column is the symbol name.

To sort the output from the largest number of samples to the smallest (reverse order), use `-r` in conjunction with the `-l` option.

-i symbol-name

List sample data specific to a symbol name. For example, running:


```
~]# oprofile -l -i __gconv_transform_utf8_internal /lib/tls/libc-version.so
```

returns the following output:

```
samples % symbol name
12 100.000 __gconv_transform_utf8_internal
```

The first line is a summary for the symbol/executable combination.

The first column is the number of samples for the memory symbol. The second column is the percentage of samples for the memory address relative to the total number of samples for the symbol. The third column is the symbol name.

-d

This option lists sample data by symbols with more detail than the **-l** option. For example, with the following command:

```
~]# oprofile -d -i __gconv_transform_utf8_internal /lib/tls/libc-version.so
```

this output is returned:

```
vma samples % symbol name
00a98640 12 100.000 __gconv_transform_utf8_internal
00a98640 1 8.3333
00a9868c 2 16.6667
00a9869a 1 8.3333
00a986c1 1 8.3333
00a98720 1 8.3333
00a98749 1 8.3333
00a98753 1 8.3333
00a98789 1 8.3333
00a98864 1 8.3333
00a98869 1 8.3333
00a98b08 1 8.3333
```

The data is the same as the **-l** option except that for each symbol, each virtual memory address used is shown. For each virtual memory address, the number of samples and percentage of samples relative to the number of samples for the symbol is displayed.

-e *symbol-name...*

With this option, you can exclude some symbols from the output. Replace *symbol-name* with the comma-separated list of symbols you want to exclude.

session:*name*

Here, you can specify the full path to the session, a directory relative to the **/var/lib/oprofile/samples/** directory, or if you are using **operf**, a directory relative to **./oprofile_data/samples/**.

20.6.3. Getting More Detailed Output on the Modules

OProfile collects data on a system-wide basis for kernel- and user-space code running on the machine. However, once a module is loaded into the kernel, the information about the origin of the kernel module is lost. The module could come from the **initrd** file on boot up, the directory with the various kernel modules, or a locally created kernel module. As a result, when OProfile records samples for a module, it just lists the samples for the modules for an executable in the root directory,

but this is unlikely to be the place with the actual code for the module. You will need to take some steps to make sure that analysis tools get the proper executable.

To get a more detailed view of the actions of the module, you will need to either have the module "unstripped" (that is installed from a custom build) or have the *debuginfo* package installed for the kernel.

Find out which kernel is running with the **uname -a** command, obtain the appropriate *debuginfo* package and install it on the machine.

Then proceed with clearing out the samples from previous runs with the following command:

```
~]# opcontrol --reset
```

To start the monitoring process, for example, on a machine with Westmere processor, run the following command:

```
~]# opcontrol --setup --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:

```
~]# opreport /ext4 -l --image-path /lib/modules/`uname -r`/kernel
CPU: Intel Westmere microarchitecture, speed 2.667e+06 MHz (estimated)
Counted CPU_CLK_UNHALTED events (Clock cycles when not halted) with a unit mask of 0x00 (No
unit mask) count 500000
warning: could not check that the binary file /lib/modules/2.6.32-191.el6.x86_64/kernel/fs/
ext4/ext4.ko has not been modified since the profile was taken. Results may be inaccurate.
samples %      symbol name
1622    9.8381  ext4_iget
1591    9.6500  ext4_find_entry
1231    7.4665  __ext4_get_inode_loc
783     4.7492  ext4_ext_get_blocks
752     4.5612  ext4_check_dir_entry
644     3.9061  ext4_mark_inode_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
```

20.6.4. Using opannotate

The **opannotate** tool tries to match the samples for particular instructions to the corresponding lines in the source code. The resulting generated files 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 Fedora, the *debuginfo* packages are not automatically installed with the corresponding packages that contain the executable. You have to obtain and install them separately.

The general syntax for **opannotate** is as follows:

```
~]# opannotate --search-dirs src-dir --source executable
```

These command-line options are mandatory. Replace *src-dir* with a path to the directory containing the source code and specify the executable to be analyzed. See the `opannotate(1)` manual page for a list of additional command line options.

20.7. Understanding the /dev/oprofile/ directory

When using OProfile in legacy mode, the **/dev/oprofile/** directory is used to store the file system for OProfile. On the other hand, **operf** does not require **/dev/oprofile/**. Use the **cat** command to display the values of the virtual files in this file system. For example, the following command displays the type of processor OProfile detected:

```
~]# cat /dev/oprofile/cpu_type
```

A directory exists in **/dev/oprofile/** for each counter. For example, if there are 2 counters, the directories **/dev/oprofile/0/** and **/dev/oprofile/1/** exist.

Each directory for a counter contains the following files:

- **count** — The interval between samples.
- **enabled** — If 0, the counter is off and no samples are collected for it; if 1, the counter is on and samples are being collected for it.
- **event** — The event to monitor.
- **extra** — Used on machines with Nehalem processors to further specify the event to monitor.
- **kernel** — If 0, samples are not collected for this counter event when the processor is in kernel-space; if 1, samples are collected even if the processor is in kernel-space.
- **unit_mask** — Defines which unit masks are enabled for the counter.
- **user** — If 0, samples are not collected for the counter event when the processor is in user-space; if 1, samples are collected even if the processor is in user-space.

The values of these files can be retrieved with the **cat** command. For example:

```
~]# cat /dev/oprofile/0/count
```

20.8. Example Usage

While OProfile can be used by developers to analyze application performance, it can also be used by system administrators to perform system analysis. For example:

- *Determine which applications and services are used the most on a system* — **opreport** can be used to determine how much processor time an application or service uses. If the system is used for multiple services but is underperforming, the services consuming the most processor time can be moved to dedicated systems.
- *Determine processor usage* — The **CPU_CLK_UNHALTED** event can be monitored to determine the processor load over a given period of time. This data can then be used to determine if additional processors or a faster processor might improve system performance.

20.9. OProfile Support for Java

OProfile allows you to profile dynamically compiled code (also known as "just-in-time" or JIT code) of the Java Virtual Machine (JVM). OProfile in Fedora 26 includes built-in support for the JVM Tools Interface (JVMTI) agent library, which supports Java 1.5 and higher.

20.9.1. Profiling Java Code

To profile JIT code from the Java Virtual Machine with the JVMTI agent, add the following to the JVM startup parameters:

```
-agentlib:jvmti_oprofile
```

Where *jvmti_oprofile* is a path to the OProfile agent. For 64-bit JVM, the path looks as follows:

```
-agentlib:/usr/lib64/oprofile/libjvmti_oprofile.so
```

Currently, you can add one command-line option: **-debug**, which enables debugging mode.



Install the oprofile-jit package

The *oprofile-jit* package must be installed on the system in order to profile JIT code with OProfile. With this package, you gain the capability to show method-level information.

Depending on the JVM that you are using, you may have to install the *debuginfo* package for the JVM. For OpenJDK, this package is required, there is no debuginfo package for Oracle JDK. To keep the debug information packages synchronized with their respective non-debug packages, you also need to install the *yum-plugin-auto-update-debug-info* plug-in. This plug-in searches the debug information repository for corresponding updates.

After successful setup, you can apply the standard profiling and analyzing tools described in previous sections

To learn more about Java support in OProfile, see the OProfile Manual, which is linked from [Section 20.12, “Additional Resources”](#).

20.10. Graphical Interface

Some OProfile preferences can be set with a graphical interface. Make sure you have the **oprofile-gui** package that provides the OProfile GUI installed on your system. To start the interface, execute the **oprof_start** command as root at a shell prompt.

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. Exiting the application does not stop OProfile from sampling.

On the **Setup** tab, to set events for the processor counters as discussed in [Section 20.3.2, “Setting Events to Monitor”](#), select the counter from the pulldown menu and select the event from the list. A brief description of the event appears in the text box below the list. Only events available for the specific counter and the specific architecture are displayed. The interface also displays whether the profiler is running and some brief statistics about it.

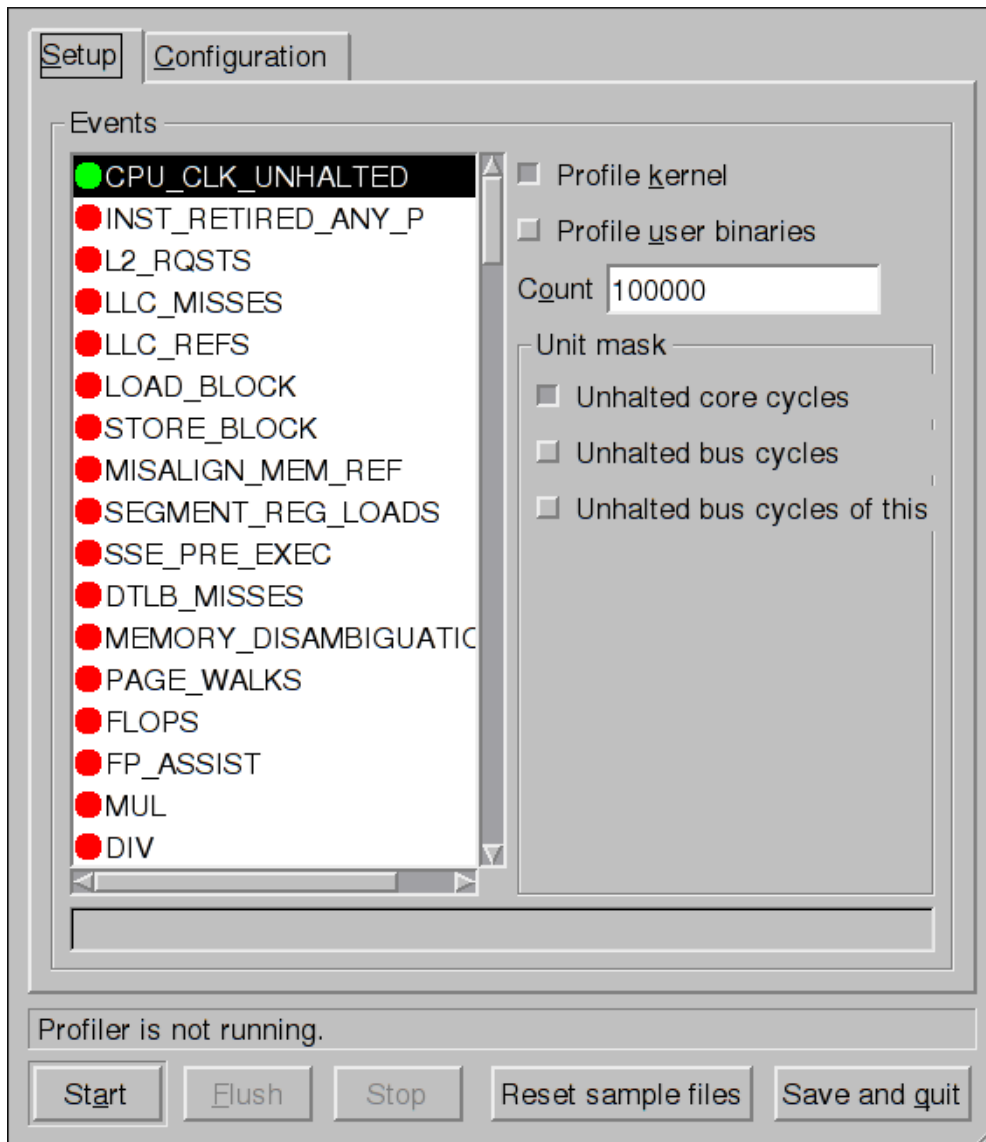


Figure 20.1. OProfile Setup

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 20.3.3, “Separating Kernel and User-space Profiles”](#). If this option is not selected, no samples are collected for the kernel.

Select the **Profile user binaries** option to count events in user mode for the currently selected event, as discussed in [Section 20.3.3, “Separating Kernel and User-space Profiles”](#). If this option is not selected, no samples are collected for user applications.

Use the **Count** text field to set the sampling rate for the currently selected event as discussed in [Section 20.3.2.1, “Sampling Rate”](#).

If any unit masks are available for the currently selected event, as discussed in [Section 20.3.2.2, “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**.

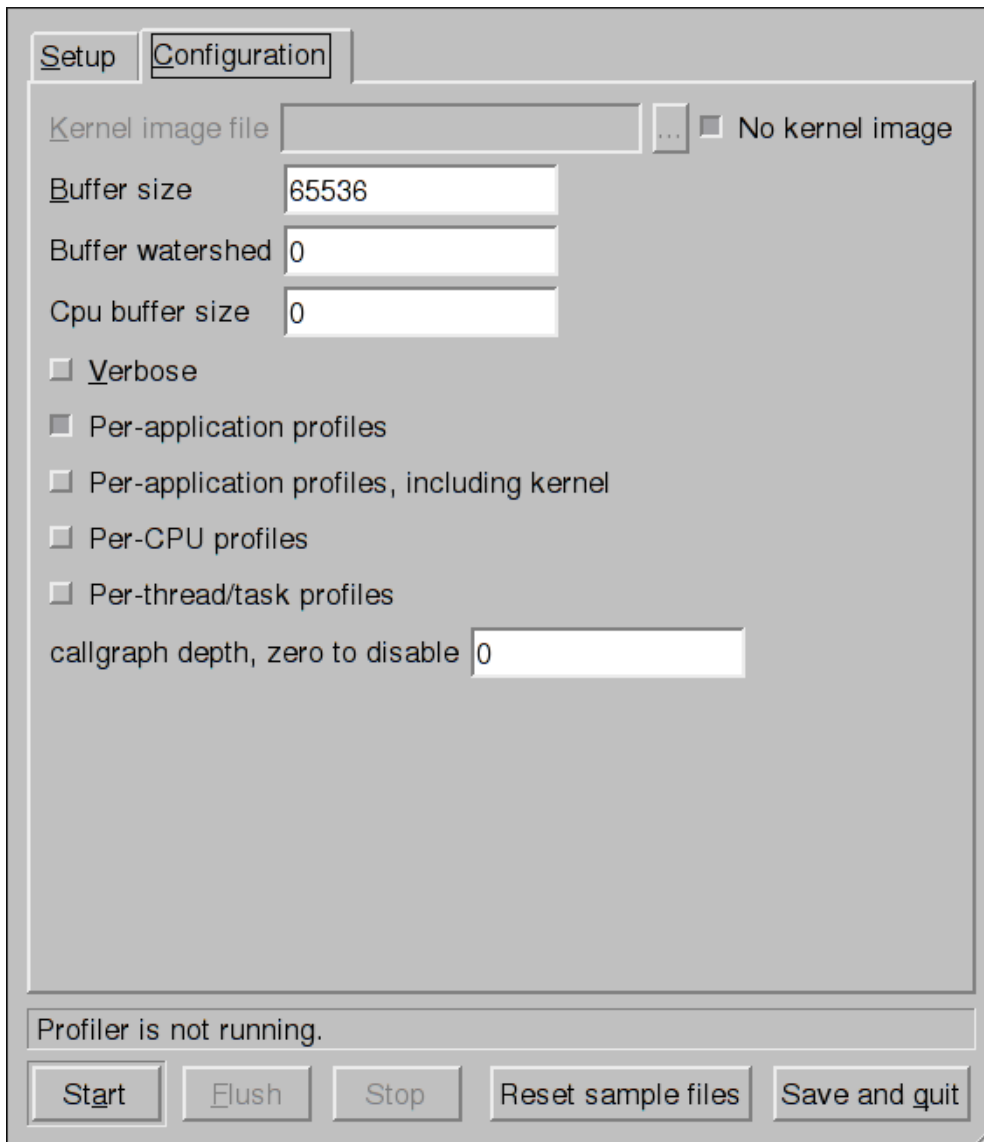


Figure 20.2. OProfile Configuration

If the **Verbose** option is selected, the **oprofiled** daemon log includes more detailed 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 20.3.3, “Separating Kernel and User-space Profiles”](#). This is equivalent to the **opcontrol --separate=kernel** command.

To force data to be written to samples files as discussed in [Section 20.6, “Analyzing the Data”](#), click the **Flush** button. This is equivalent to the **opcontrol --dump** command.

To start OProfile from the graphical interface, click **Start**. To stop the profiler, click **Stop**. Exiting the application does not stop OProfile from sampling.

20.11. OProfile and SystemTap

SystemTap is a tracing and probing tool that allows users to study and monitor the activities of the operating system in fine detail. It provides information similar to the output of tools like **netstat**, **ps**,

top, and **iostat**; however, SystemTap is designed to provide more filtering and analysis options for the collected information.

While using OProfile is suggested in cases of collecting data on where and why the processor spends time in a particular area of code, it is less usable when finding out why the processor stays idle.

You might want to use SystemTap when instrumenting specific places in code. Because SystemTap allows you to run the code instrumentation without having to stop and restart the instrumented code, it is particularly useful for instrumenting the kernel and daemons.

For more information on SystemTap, see [the section called “Online Documentation”](#) for the relevant SystemTap documentation.

20.12. Additional Resources

To learn more about OProfile and how to configure it, see the following resources.

Installed Documentation

- `/usr/share/doc/oprofile/oprofile.html` — *OProfile Manual*
- **oprofile(1)** manual page — Discusses **opcontrol**, **opreport**, **opannotate**, and **ophelp**
- **operf(1)** manual page

Online Documentation

- <http://oprofile.sourceforge.net/> — Contains the latest upstream documentation, mailing lists, IRC channels, and more.

See Also

- [SystemTap Beginners Guide](#)¹ — Provides basic instructions on how to use SystemTap to monitor different subsystems of Fedora in finer detail.

¹ https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/SystemTap_Beginners_Guide/index.html

Part VI. Kernel, Module and Driver Configuration

This part covers various tools that assist administrators with kernel customization.

Working with the GRUB 2 Boot Loader

Fedora 26 is distributed with the GNU GRand Unified Boot loader (GRUB) version 2 boot loader, which allows the user to select an operating system or kernel to be loaded at system boot time. GRUB 2 also allows the user to pass arguments to the kernel.

21.1. Introduction to GRUB 2

GRUB 2 reads its configuration from the `/boot/grub2/grub.cfg` file on traditional BIOS-based machines and from the `/boot/efi/EFI/fedora/grub.cfg` file on UEFI machines. This file contains menu information.

The GRUB 2 configuration file, `grub.cfg`, is generated during installation, or by invoking the `usr/sbin/grub2-mkconfig` utility, and is automatically updated by `grubby` each time a new kernel is installed. When regenerated manually using `grub2-mkconfig`, the file is generated according to the template files located in `/etc/grub.d/`, and custom settings in the `/etc/default/grub` file. Edits of `grub.cfg` will be lost any time `grub2-mkconfig` is used to regenerate the file, so care must be taken to reflect any manual changes in `/etc/default/grub` as well.

Normal operations on `grub.cfg`, such as the removal and addition of new kernels, should be done using the `grubby` tool and, for scripts, using `new-kernel-pkg` tool. If you use `grubby` to modify the default kernel the changes will be inherited when new kernels are installed. For more information on `grubby`, see [Section 21.4, “Making Persistent Changes to a GRUB 2 Menu Using the grubby Tool”](#).

The `/etc/default/grub` file is used by the `grub2-mkconfig` tool, which is used by anaconda when creating `grub.cfg` during the installation process, and can be used in the event of a system failure, for example if the boot loader configurations need to be recreated. In general, it is not recommended to replace the `grub.cfg` file by manually running `grub2-mkconfig` except as a last resort. Note that any manual changes to `/etc/default/grub` require rebuilding the `grub.cfg` file.

Menu Entries in grub.cfg

Among various code snippets and directives, the `grub.cfg` configuration file contains one or more `menuentry` blocks, each representing a single GRUB 2 boot menu entry. These blocks always start with the `menuentry` keyword followed by a title, list of options, and an opening curly bracket, and end with a closing curly bracket. Anything between the opening and closing bracket should be indented. For example, the following is a sample `menuentry` block for Fedora 26 with Linux kernel 3.17.4-301.fc21.x86_64:

```
menuentry 'Fedora, with Linux 3.17.4-301.fc21.x86_64' --class fedora --class gnu-linux --
class gnu --class os --unrestricted $menuentry_id_option 'gnulinux-3.17.4-301.fc21.x86_64-
advanced-effee860-8d55-4e4a-995e-b4c88f9ac9f0' {
    load_video
    set gfxpayload=keep
    insmod gzio
    insmod part_msdos
    insmod ext2
    set root='hd0,msdos1'
    if [ x$feature_platform_search_hint = xy ]; then
        search --no-floppy --fs-uuid --set=root --hint='hd0,msdos1' f19c92f4-9ead-4207-
b46a-723b7a2c51c8
    else
        search --no-floppy --fs-uuid --set=root f19c92f4-9ead-4207-b46a-723b7a2c51c8
    fi
    linux16 /vmlinuz-3.17.4-301.fc21.x86_64 root=/dev/mapper/fedora-root ro
    rd.lvm.lv=fedora/swap rd.lvm.lv=fedora/root rhgb quiet LANG=en_US.UTF-8
    initrd16 /initramfs-3.17.4-301.fc21.x86_64.img
```



```
}
```

Each **menuentry** block that represents an installed Linux kernel contains **linux** on 64-bit IBM POWER Series, **linux16** on x86_64 BIOS-based systems, and **linuxefi** on UEFI-based systems. Then the **initrd** directives followed by the path to the kernel and the **initramfs** image respectively. If a separate **/boot** partition was created, the paths to the kernel and the **initramfs** image are relative to **/boot**. In the example above, the **initrd /initramfs-3.17.4-301.fc21.x86_64.img** line means that the **initramfs** image is actually located at **/boot/initramfs-3.17.4-301.fc21.x86_64.img** when the root file system is mounted, and likewise for the kernel path.

The kernel version number as given on the **linux16 /vmlinuz-kernel_version** line must match the version number of the **initramfs** image given on the **initrd /initramfs-kernel_version.img** line of each **menuentry** block. For more information on how to verify the initial RAM disk image, see [Section 22.5, “Verifying the Initial RAM Disk Image”](#).

Note

In **menuentry** blocks, the **initrd** directive must point to the location (relative to the **/boot/** directory if it is on a separate partition) of the **initramfs** file corresponding to the same kernel version. This directive is called **initrd** because the previous tool which created initial RAM disk images, **mkinitrd**, created what were known as **initrd** files. The **grub.cfg** directive remains **initrd** to maintain compatibility with other tools. The file-naming convention of systems using the **dracut** utility to create the initial RAM disk image is **initramfs-kernel_version.img**.

For information on using **Dracut**, see [Section 22.5, “Verifying the Initial RAM Disk Image”](#).

21.2. Configuring the GRUB 2 Boot Loader

Changes to the GRUB 2 menu can be made temporarily at boot time, made persistent for a single system while the system is running, or as part of making a new GRUB 2 configuration file.

- To make non-persistent changes to the GRUB 2 menu, see [Section 21.3, “Making Temporary Changes to a GRUB 2 Menu”](#).
- To make persistent changes to a running system, see [Section 21.4, “Making Persistent Changes to a GRUB 2 Menu Using the grubby Tool”](#).
- For information on making and customizing a GRUB 2 configuration file, see [Section 21.5, “Customizing the GRUB 2 Configuration File”](#).

21.3. Making Temporary Changes to a GRUB 2 Menu

Procedure 21.1. Making Temporary Changes to a Kernel Menu Entry

To change kernel parameters only during a single boot process, proceed as follows:

1. Start the system and, on the GRUB 2 boot screen, move the cursor to the menu entry you want to edit, and press the **e** key for edit.
2. Move the cursor down to find the kernel command line. The kernel command line starts with **linux** on 64-Bit IBM Power Series, **linux16** on x86-64 BIOS-based systems, or **linuxefi** on UEFI systems.
3. Move the cursor to the end of the line.

Press **Ctrl+a** and **Ctrl+e** to jump to the start and end of the line, respectively. On some systems, **Home** and **End** might also work.

4. Edit the kernel parameters as required. For example, to run the system in emergency mode, add the *emergency* parameter at the end of the **linux16** line:

```
linux16      /vmlinuz-4.2.0-1.fc23.x86_64 root=/dev/mapper/fedora-root ro rd.md=0
rd.dm=0 rd.lvm.lv=fedora/swap crashkernel=auto rd.luks=0 vconsole.keymap=us
rd.lvm.lv=fedora/root rhgb quiet emergency
```

The **rhgb** and **quiet** parameters can be removed in order to enable system messages.

These settings are not persistent and apply only for a single boot. To make persistent changes to a menu entry on a system, use the **grubby** tool. See [the section called “Adding and Removing Arguments from a GRUB Menu Entry”](#) for more information on using **grubby**.

21.4. Making Persistent Changes to a GRUB 2 Menu Using the grubby Tool

The **grubby** tool can be used to read information from, and make persistent changes to, the **grub.cfg** file. It enables, for example, changing GRUB menu entries to specify what arguments to pass to a kernel on system start and changing the default kernel.

In Red Hat Enterprise Linux 7, if **grubby** is invoked manually without specifying a GRUB configuration file, it defaults to searching for **/etc/grub2.cfg**, which is a symbolic link to the **grub.cfg** file, whose location is architecture dependent. If that file cannot be found it will search for an architecture dependent default.

Listing the Default Kernel

To find out the file name of the default kernel, enter a command as follows:

```
~]# grubby --default-kernel
/boot/vmlinuz-4.2.0-1.fc23.x86_64
```

To find out the index number of the default kernel, enter a command as follows:

```
~]# grubby --default-index
0
```

Changing the Default Boot Entry

To make a persistent change in the kernel designated as the default kernel, use the **grubby** command as follows:

```
~]# grubby --set-default /boot/vmlinuz-4.2.0-1.fc23.x86_64
```

Viewing the GRUB Menu Entry for a Kernel

To list all the kernel menu entries, enter a command as follows:

```
~]$ grubby --info=ALL
```

On UEFI systems, all **grubby** commands must be entered as root.

To view the GRUB menu entry for a specific kernel, enter a command as follows:

```
~]$ grubby --info /boot/vmlinuz-4.2.0-1.fc23.x86_64
index=0
kernel=/boot/vmlinuz-4.2.0-1.fc23.x86_64
args="ro rd.lvm.lv=fedora/root rd.lvm.lv=fedora/swap rhgb quiet LANG=en_US.UTF-8"
root=/dev/mapper/fedora-root
initrd=/boot/initramfs-4.2.0-1.fc23.x86_64.img
title=Fedora (4.2.0-1.fc23.x86_64) 23 (Workstation Edition)
```

Try tab completion to see the available kernels within the **/boot/** directory.

Adding and Removing Arguments from a GRUB Menu Entry

The **--update-kernel** option can be used to update a menu entry when used in combination with **--args** to add new arguments and **--remove-arguments** to remove existing arguments. These options accept a quoted space-separated list. The command to simultaneously add and remove arguments from a GRUB menu entry has the following format:

```
grubby --remove-args="argX argY" --args="argA argB" --update-kernel /boot/kernel
```

To add and remove arguments from a kernel's GRUB menu entry, use a command as follows:

```
~]# grubby --remove-args="rhgb quiet" --args=console=ttyS0,115200 --update-kernel /boot/vmlinuz-4.2.0-1.fc23.x86_64
```

This command removes the Red Hat graphical boot argument, enables boot message to be seen, and adds a serial console. As the console arguments will be added at the end of the line, the new console will take precedence over any other consoles configured.

To review the changes, use the **--info** command option as follows:

```
~]# grubby --info /boot/vmlinuz-4.2.0-1.fc23.x86_64
index=0
kernel=/boot/vmlinuz-4.2.0-1.fc23.x86_64
args="ro rd.lvm.lv=fedora/root rd.lvm.lv=fedora/swap LANG=en_US.UTF-8 console=ttyS0,115200"
root=/dev/mapper/fedora-root
initrd=/boot/initramfs-4.2.0-1.fc23.x86_64.img
title=Fedora (4.2.0-1.fc23.x86_64) 23 (Workstation Edition)
```

Updating All Kernel Menus with the Same Arguments

To add the same kernel boot arguments to all the kernel menu entries, enter a command as follows:

```
~]# grubby --update-kernel=ALL --args=console=ttyS0,115200
```

The **--update-kernel** parameter also accepts **DEFAULT** or a comma separated list of kernel index numbers.

Changing a Kernel Argument

To change a value in an existing kernel argument, specify the argument again, changing the value as required. For example, if the virtual console font size has been set to **latarcyrheb-sun16** and you want to change the virtual console font size to **32**, use a command as follows:

```
~]# grubby --args=vconsole.font=latarcyrheb-sun32 --update-kernel /boot/vmlinuz-4.2.0-1.fc23.x86_64
```



```

index=0
kernel=/boot/vmlinuz-4.2.0-1.fc23.x86_64
args="ro rd.lvm.lv=fedora/root crashkernel=auto rd.lvm.lv=fedora/swap
vconsole.font=latarcyrheb-sun32 vconsole.keymap=us LANG=en_US.UTF-8"
root=/dev/mapper/fedora-root
initrd=/boot/initramfs-4.2.0-1.fc23.x86_64.img
title=Fedora (4.2.0-1.fc23.x86_64) 23 (Workstation Edition)

```

See the **grubby(8)** manual page for more command options.

21.5. Customizing the GRUB 2 Configuration File

GRUB 2 scripts search the user's computer and build a boot menu based on what operating systems the scripts find. To reflect the latest system boot options, the boot menu is rebuilt automatically when the kernel is updated or a new kernel is added.

However, users may want to build a menu containing specific entries or to have the entries in a specific order. GRUB 2 allows basic customization of the boot menu to give users control of what actually appears on the screen.

GRUB 2 uses a series of scripts to build the menu; these are located in the `/etc/grub.d/` directory. The following files are included:

- **00_header**, which loads GRUB 2 settings from the `/etc/default/grub` file.
- **01_users**, which is created only when a boot loader password is assigned in a **kickstart** file.
- **10_linux**, which locates kernels in the default partition of Fedora.
- **30_os-prober**, which builds entries for operating systems found on other partitions.
- **40_custom**, a template, which can be used to create additional menu entries.

Scripts from the `/etc/grub.d/` directory are read in alphabetical order and can be therefore renamed to change the boot order of specific menu entries.



Important

With the **GRUB_TIMEOUT** key set to **0** in the `/etc/default/grub` file, GRUB 2 does not display the list of bootable kernels when the system starts up. In order to display this list when booting, press and hold any alphanumeric key when the BIOS information is displayed; GRUB 2 will present you with the GRUB menu.

21.5.1. Changing the Default Boot Entry

By default, the key for the **GRUB_DEFAULT** directive in the `/etc/default/grub` file is the word **saved**. This instructs GRUB 2 to load the kernel specified by the **saved_entry** directive in the GRUB 2 environment file, located at `/boot/grub2/grubenv`. You can set another GRUB record to be the default, using the **grub2-set-default** command, which will update the GRUB 2 environment file.

By default, the **saved_entry** value is set to the name of latest installed kernel of package type *kernel*. This is defined in `/etc/sysconfig/kernel` by the **UPDATEDEFAULT** and **DEFAULTKERNEL** directives. The file can be viewed by the root user as follows:


```
~]# cat /etc/sysconfig/kernel
# UPDATEDEFAULT specifies if new-kernel-pkg should make
# new kernels the default
UPDATEDEFAULT=yes

# DEFAULTKERNEL specifies the default kernel package type
DEFAULTKERNEL=kernel-core
```

The **DEFAULTKERNEL** directive specifies what package type will be used as the default. Installing a package of type *kernel-debug* will not change the default kernel while the **DEFAULTKERNEL** is set to package type *kernel*.

GRUB 2 supports using a numeric value as the key for the **saved_entry** directive to change the default order in which the operating systems are loaded. To specify which operating system should be loaded first, pass its number to the **grub2-set-default** command. For example:

```
~]# grub2-set-default 2
```

Note that the position of a menu entry in the list is denoted by a number starting with zero; therefore, in the example above, the third entry will be loaded. This value will be overwritten by the name of the next kernel to be installed.

To force a system to always use a particular menu entry, use the menu entry name as the key to the **GRUB_DEFAULT** directive in the **/etc/default/grub** file. To list the available menu entries, run the following command as root:

```
~]# awk -F\' ' $1=="menuentry " {print $2}' /etc/grub2.cfg
```

The file name **/etc/grub2.cfg** is a symlink to the **grub.cfg** file, whose location is architecture dependent. For reliability reasons, the symlink is not used in other examples in this chapter. It is better to use absolute paths when writing to a file, especially when repairing a system.

Changes to **/etc/default/grub** require rebuilding the **grub.cfg** file as follows:

- On BIOS-based machines, issue the following command as root:

```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

- On UEFI-based machines, issue the following command as root:

```
~]# grub2-mkconfig -o /boot/efi/EFI/fedora/grub.cfg
```

21.5.2. Editing a Menu Entry

If required to prepare a new GRUB 2 file with different parameters, edit the values of the **GRUB_CMDLINE_LINUX** key in the **/etc/default/grub** file. Note that you can specify multiple parameters for the **GRUB_CMDLINE_LINUX** key. For example:

```
GRUB_CMDLINE_LINUX="console=tty0 console=ttyS0,9600n8"
```

Where **console=tty0** is the first virtual terminal and **console=ttyS0** is the serial terminal to be used.

Changes to **/etc/default/grub** require rebuilding the **grub.cfg** file as follows:

- On BIOS-based machines, issue the following command as root:


```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

- On UEFI-based machines, issue the following command as root:

```
~]# grub2-mkconfig -o /boot/efi/EFI/fedora/grub.cfg
```

21.5.3. Adding a new Entry

When executing the **grub2-mkconfig** command, GRUB 2 searches for Linux kernels and other operating systems based on the files located in the **/etc/grub.d/** directory. The **/etc/grub.d/10_linux** script searches for installed Linux kernels on the same partition. The **/etc/grub.d/30_os-prober** script searches for other operating systems. Menu entries are also automatically added to the boot menu when updating the kernel.

The **40_custom** file located in the **/etc/grub.d/** directory is a template for custom entries and looks as follows:

```
#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries.  Simply type the
# menu entries you want to add after this comment.  Be careful not to change
# the 'exec tail' line above.
```

This file can be edited or copied. Note that as a minimum, a valid menu entry must include at least the following:

```
menuentry "<Title>"{
<Data>
}
```

21.5.4. Creating a Custom Menu

If you do not want menu entries to be updated automatically, you can create a custom menu.



Important

Before proceeding, back up the contents of the **/etc/grub.d/** directory in case you need to revert the changes later.



Note

Note that modifying the **/etc/default/grub** file does not have any effect on creating custom menus.

1. On BIOS-based machines, copy the contents of **/boot/grub2/grub.cfg**, or, on UEFI machines, copy the contents of **/boot/efi/EFI/fedora/grub.cfg**. Put the content of

the **grub.cfg** into the **/etc/grub.d/40_custom** file below the existing header lines. The executable part of the **40_custom** script has to be preserved.

- From the content put into the **/etc/grub.d/40_custom** file, only the **menuentry** blocks are needed to create the custom menu. The **/boot/grub2/grub.cfg** and **/boot/efi/EFI/fedora/grub.cfg** files might contain function specifications and other content above and below the **menuentry** blocks. If you put these unnecessary lines into the **40_custom** file in the previous step, erase them.

This is an example of a custom **40_custom** script:

```
#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries.  Simply type the
# menu entries you want to add after this comment.  Be careful not to change
# the 'exec tail' line above.

menuentry 'First custom entry' --class red --class gnu-linux --class gnu --class
os $menuentry_id_option 'gnulinux-4.2.0-1.fc23.x86_64-advanced-32782dd0-4b47-4d56-
a740-2076ab5e5976' {
    load_video
    set gfxpayload=keep
    insmod gzio
    insmod part_msdos
    insmod xfs
    set root='hd0,msdos1'
    if [ x$feature_platform_search_hint = xy ]; then
        search --no-floppy --fs-uuid --set=root --hint='hd0,msdos1'
7885bba1-8aa7-4e5d-a7ad-821f4f52170a
    else
        search --no-floppy --fs-uuid --set=root 7885bba1-8aa7-4e5d-a7ad-821f4f52170a
    fi
    linux16 /vmlinuz-4.2.0-1.fc23.x86_64 root=/dev/mapper/fedora-root ro
rd.lvm.lv=fedora/root vconsole.font=latacyrheb-sun16 rd.lvm.lv=fedora/swap
vconsole.keymap=us crashkernel=auto rhgb quiet LANG=en_US.UTF-8
    initrd16 /initramfs-4.2.0-1.fc23.x86_64.img
}
menuentry 'Second custom entry' --class red --class gnu-linux --class gnu --class
os $menuentry_id_option 'gnulinux-0-rescue-07f43f20a54c4ce8ada8b70d33fd001c-
advanced-32782dd0-4b47-4d56-a740-2076ab5e5976' {
    load_video
    insmod gzio
    insmod part_msdos
    insmod xfs
    set root='hd0,msdos1'
    if [ x$feature_platform_search_hint = xy ]; then
        search --no-floppy --fs-uuid --set=root --hint='hd0,msdos1'
7885bba1-8aa7-4e5d-a7ad-821f4f52170a
    else
        search --no-floppy --fs-uuid --set=root 7885bba1-8aa7-4e5d-a7ad-821f4f52170a
    fi
    linux16 /vmlinuz-0-rescue-07f43f20a54c4ce8ada8b70d33fd001c root=/dev/mapper/
fedora-root ro rd.lvm.lv=fedora/root vconsole.font=latacyrheb-sun16 rd.lvm.lv=fedora/
swap vconsole.keymap=us crashkernel=auto rhgb quiet
    initrd16 /initramfs-0-rescue-07f43f20a54c4ce8ada8b70d33fd001c.img
}
```

- Remove all files from the **/etc/grub.d** directory except the following:

- **00_header**,
- **40_custom**,

- **01_users** (if it exists),
- and **README**.

Alternatively, if you want to keep the files in the **/etc/grub2.d/** directory, make them unexecutable by running the **chmod a-x <file_name>** command.

4. Edit, add, or remove menu entries in the **40_custom** file as desired.
5. Rebuild the **grub.cfg** file by running the **grub2-mkconfig -o** command as follows:
 - On BIOS-based machines, issue the following command as root:

```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

- On UEFI-based machines, issue the following command as root:

```
~]# grub2-mkconfig -o /boot/efi/EFI/fedora/grub.cfg
```

21.6. GRUB 2 Password Protection

GRUB 2 supports both plain-text and encrypted passwords in the GRUB 2 template files. To enable the use of passwords, specify a superuser who can reach the protected entries. Other users can be specified to access these entries as well. Menu entries can be password-protected for booting by adding one or more users to the menu entry as described in [Section 21.6.1, “Setting Up Users and Password Protection, Specifying Menu Entries”](#). To use encrypted passwords, see [Section 21.6.2, “Password Encryption”](#).



Warning

If you do not use the correct format for the menu, or modify the configuration in an incorrect way, you might be unable to boot your system.

All menu entries can be password-protected against changes by setting superusers, which can be done in the **/etc/grub.d/00_header** or the **/etc/grub.d/01_users** file. The **00_header** file is very complicated and, if possible, avoid making modifications in this file. Menu entries should be placed in the **/etc/grub.d/40_custom** and users in the **/etc/grub.d/01_users** file. The **01_users** file is generated by the installation application **anaconda** when a grub boot loader password is used in a **kickstart** template (but it should be created and used if it does not exist). Examples in this section adopt this policy.

21.6.1. Setting Up Users and Password Protection, Specifying Menu Entries

1. To specify a superuser, add the following lines in the **/etc/grub.d/01_users** file, where **john** is the name of the user designated as the superuser, and **johnspassword** is the superuser's password:

```
cat <<EOF
set superusers="john"
```



```
password john johnspassword
EOF
```

2. To allow other users to access the menu entries, add additional lines per user at the end of the `/etc/grub.d/01_users` file.

```
cat <<EOF
set superusers="john"
password john johnspassword
password jane janespassword
EOF
```

3. When the users and passwords are set up, specify the menu entries that should be password-protected in the `/etc/grub.d/40_custom` file in a similar fashion to the following:

```
menuentry 'Red Hat Enterprise Linux Server' --unrestricted {
set root=(hd0,msdos1)
linux /vmlinuz
}

menuentry 'Fedora' --users jane {
set root=(hd0,msdos2)
linux /vmlinuz
}

menuentry 'Red Hat Enterprise Linux Workstation' {
set root=(hd0,msdos3)
linux /vmlinuz
}
```

In the above example:

- john is the **superuser** and can therefore boot any menu entry, use the GRUB 2 command line, and edit items of the GRUB 2 menu during boot. In this case, john can access both Red Hat Enterprise Linux Server, Fedora, and Red Hat Enterprise Linux Workstation. Note that only john can access Red Hat Enterprise Linux Workstation because neither the **--users** nor **--unrestricted** options have been used.
- User jane can boot Fedora since she was granted the permission in the configuration.
- Anyone can boot Red Hat Enterprise Linux Server, because of the **--unrestricted** option, but only john can edit the menu entry as a superuser has been defined. When a superuser is defined then all records are protected against unauthorized changes and all records are protected for booting if they do **not** have the **--unrestricted** parameter

If you do not specify a user for a menu entry, or make use of the **--unrestricted** option, then only the superuser will have access to the system.

After you have made changes in the template file the GRUB 2 configuration file must be updated.

Rebuild the **grub.cfg** file by running the **grub2-mkconfig -o** command as follows:

- On BIOS-based machines, issue the following command as root:

```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

- On UEFI-based machines, issue the following command as root:


```
~]# grub2-mkconfig -o /boot/efi/EFI/fedora/grub.cfg
```

21.6.2. Password Encryption

By default, passwords are saved in plain text in GRUB 2 scripts. Although the files cannot be accessed on boot without the correct password, security can be improved by encrypting the password using the **grub2-mkpasswd-pbkdf2** command. This command converts a desired password into a long hash, which is placed in the GRUB 2 scripts instead of the plain-text password.

1. To generate an encrypted password, run the **grub2-mkpasswd-pbkdf2** command on the command line as root.
2. Enter the desired password when prompted and repeat it. The command then outputs your password in an encrypted form.
3. Copy the hash, and paste it in the template file where you configured the users, that is, either in **/etc/grub.d/01_users** or **/etc/grub.d/40_custom**.

The following format applies for the **01_users** file:

```
cat <<EOF
set superusers="john"
password_pbkdf2 john
  grub.pbkdf2.sha512.10000.19074739ED80F115963D984BDCB35AA671C24325755377C3E9B014D862DA6ACC77BC110EED
EOF
```

The following format applies for the **40_custom** file:

```
set superusers="john"
password_pbkdf2 john
  grub.pbkdf2.sha512.10000.19074739ED80F115963D984BDCB35AA671C24325755377C3E9B014D862DA6ACC77BC110EED
```

21.7. Reinstalling GRUB 2

Reinstalling GRUB 2 is a convenient way to fix certain problems usually caused by an incorrect installation of GRUB 2, missing files, or a broken system. Other reasons to reinstall GRUB 2 include the following:

- Upgrading from the previous version of GRUB.
- The user requires the GRUB 2 boot loader to control installed operating systems. However, some operating systems are installed with their own boot loaders. Reinstalling GRUB 2 returns control to the desired operating system.
- Adding the boot information to another drive.

21.7.1. Reinstalling GRUB 2 on BIOS-Based Machines

When using the **grub2-install** command, the boot information is updated and missing files are restored. Note that the files are restored only if they are not corrupted.

Use the **grub2-install device** command to reinstall GRUB 2 if the system is operating normally. For example, if **sda** is your *device*:

```
~]# grub2-install /dev/sda
```


21.7.2. Reinstalling GRUB 2 on UEFI-Based Machines

When using the `dnf reinstall grub2-efi shim` command, the boot information is updated and missing files are restored. Note that the files are restored only if they are not corrupted.

Use the `dnf reinstall grub2-efi shim` command to reinstall GRUB 2 if the system is operating normally. For example:

```
~]# dnf reinstall grub2-efi shim
```

21.7.3. Resetting and Reinstalling GRUB 2

This method completely removes all GRUB 2 configuration files and system settings. Apply this method to reset all configuration settings to their default values. Removing of the configuration files and subsequent reinstalling of GRUB 2 fixes failures caused by corrupted files and incorrect configuration. To do so, as root, follow these steps:

1. Run the `rm /etc/grub.d/*` command;
2. Run the `rm /etc/sysconfig/grub` command;
3. For EFI systems **only**, run the following command:

```
~]# dnf reinstall grub2-efi shim grub2-tools
```

4. Rebuild the `grub.cfg` file by running the `grub2-mkconfig -o` command as follows:

- On BIOS-based machines, issue the following command as root:

```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

- On UEFI-based machines, issue the following command as root:

```
~]# grub2-mkconfig -o /boot/efi/EFI/fedora/grub.cfg
```

5. Now follow the procedure in [Section 21.7, “Reinstalling GRUB 2”](#) to restore GRUB2 on the `/boot/` partition.

21.8. GRUB 2 over a Serial Console

If you use computers with no display or keyboard, it can be very useful to control the machines through serial communications.

21.8.1. Configuring the GRUB 2 Menu

To set the system to use a serial terminal only during a single boot process, when the GRUB 2 boot menu appears, move the cursor to the kernel you want to start, and press the **e** key to edit the kernel parameters. Remove the **rhgb** and **quit** parameters and add console parameters at the end of the **linux16** line as follows:

```
linux16          /vmlinuz-4.2.0-1.fc23.x86_64 root=/dev/mapper/fedora-root ro rd.md=0 rd.dm=0
rd.lvm.lv=fedora/swap crashkernel=auto rd.luks=0 vconsole.keymap=us rd.lvm.lv=fedora/
root console=ttyS0,115200
```

These settings are not persistent and apply only for a single boot.

To make persistent changes to a menu entry on a system, use the **grubby** tool. For example, to update the entry for the default kernel, enter a command as follows:

```
~]# grubby --remove-args="rhgb quiet" --args=console=ttyS0,115200 --update-kernel=DEFAULT
```

The **--update-kernel** parameter also accepts the keyword **ALL** or a comma separated list of kernel index numbers. See [the section called “Adding and Removing Arguments from a GRUB Menu Entry”](#) for more information on using **grubby**.

If required to build a new GRUB 2 configuration file, add the following two lines in the **/etc/default/grub** file:

```
GRUB_TERMINAL="serial"
GRUB_SERIAL_COMMAND="serial --speed=9600 --unit=0 --word=8 --parity=no --stop=1"
```

The first line disables the graphical terminal. Note that specifying the **GRUB_TERMINAL** key overrides values of **GRUB_TERMINAL_INPUT** and **GRUB_TERMINAL_OUTPUT**. On the second line, adjust the baud rate, parity, and other values to fit your environment and hardware. A much higher baud rate, for example **115200**, is preferable for tasks such as following log files. Once you have completed the changes in the **/etc/default/grub** file, it is necessary to update the GRUB 2 configuration file.

Rebuild the **grub.cfg** file by running the **grub2-mkconfig -o** command as follows:

- On BIOS-based machines, issue the following command as root:

```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

- On UEFI-based machines, issue the following command as root:

```
~]# grub2-mkconfig -o /boot/efi/EFI/fedora/grub.cfg
```

Note

In order to access the grub terminal over a serial connection an additional option must be added to a kernel definition to make that particular kernel monitor a serial connection. For example:

```
console=ttyS0,9600n8
```

Where **console=ttyS0** is the serial terminal to be used, **9600** is the baud rate, **n** is for no parity, and **8** is the word length in bits. A much higher baud rate, for example **115200**, is preferable for tasks such as following log files.

For more information on serial console settings, see [the section called “Installable and External Documentation”](#)

21.8.2. Using screen to Connect to the Serial Console

The **screen** tool serves as a capable serial terminal. To install it, run as root:

```
~]# dnf install screen
```


To connect to your machine using the serial console, use a command in the follow format:

```
screen /dev/console_port baud_rate
```

By default, if no option is specified, **screen** uses the standard 9600 baud rate. To set a higher baud rate, enter:

```
~]$ screen /dev/console_port 115200
```

Where *console_port* is **ttys0**, or **ttys0**, and so on.

To end the session in **screen**, press **Ctrl+a**, type **:quit** and press **Enter**.

See the **screen(1)** manual page for additional options and detailed information.

21.9. Terminal Menu Editing During Boot

Menu entries can be modified and arguments passed to the kernel on boot. This is done using the menu entry editor interface, which is triggered when pressing the **e** key on a selected menu entry in the boot loader menu. The **Esc** key discards any changes and reloads the standard menu interface. The **c** key loads the command line interface.

The command line interface is the most basic GRUB interface, but it is also the one that grants the most control. The command line makes it possible to type any relevant GRUB commands followed by the **Enter** key to execute them. This interface features some advanced features similar to **shell**, including **Tab** key completion based on context, and **Ctrl+a** to move to the beginning of a line and **Ctrl+e** to move to the end of a line. In addition, the **arrow**, **Home**, **End**, and **Delete** keys work as they do in the bash shell.

21.9.1. Booting to Rescue Mode

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

1. To enter rescue mode during boot, on the GRUB 2 boot screen, press the **e** key for edit.
2. Add the following parameter at the end of the **linux** line on 64-Bit IBM Power Series, the **linux16** line on x86-64 BIOS-based systems, or the **linuxefi** line on UEFI systems:

```
systemd.unit=rescue.target
```

Press **Ctrl+a** and **Ctrl+e** to jump to the start and end of the line, respectively. On some systems, **Home** and **End** might also work.

Note that equivalent parameters, **1**, **s**, and **single**, can be passed to the kernel as well.

3. Press **Ctrl+x** to boot the system with the parameter.

21.9.2. Booting to Emergency Mode

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

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

1. To enter emergency mode, on the GRUB 2 boot screen, press the **e** key for edit.
2. Add the following parameter at the end of the **linux** line on 64-Bit IBM Power Series, the **linux16** line on x86-64 BIOS-based systems, or the **linuxefi** line on UEFI systems:

```
systemd.unit=emergency.target
```

Press **Ctrl+a** and **Ctrl+e** to jump to the start and end of the line, respectively. On some systems, **Home** and **End** might also work.

Note that equivalent parameters, *emergency* and *-b*, can be passed to the kernel as well.

3. Press **Ctrl+x** to boot the system with the parameter.

21.9.3. Changing and Resetting the Root Password

Setting up the root password is a mandatory part of the Fedora installation. If you forget or lose the root password it is possible to reset it, however users who are members of the wheel group can change the root password as follows:

```
~]$ sudo passwd root
```

Note that in GRUB 2, resetting the password is no longer performed in single-user mode as it was in GRUB included in Fedora 15 and Red Hat Enterprise Linux 6. The root password is now required to operate in **single-user** mode as well as in **emergency** mode.

Two procedures for resetting the root password are shown here:

- [Procedure 21.2, “Resetting the Root Password Using an Installation Disk”](#) takes you to a shell prompt, without having to edit the grub menu. It is the shorter of the two procedures and it is also the recommended method. You can use a server boot disk or a netinstall installation disk.
- [Procedure 21.3, “Resetting the Root Password Using rd.break”](#) makes use of **rd.break** to interrupt the boot process before control is passed from **initramfs** to **systemd**. The disadvantage of this method is that it requires more steps, includes having to edit the GRUB menu, and involves choosing between a possibly time consuming SELinux file relabel or changing the SELinux enforcing mode and then restoring the SELinux security context for **/etc/shadow/** when the boot completes.

Procedure 21.2. Resetting the Root Password Using an Installation Disk

1. Start the system and when BIOS information is displayed, select the option for a boot menu and select to boot from the installation disk.
2. Choose **Troubleshooting**.
3. Choose **Rescue a Fedora-Server System**.
4. Choose **Continue** which is the default option. At this point you will be promoted for a passphrase if an encrypted file system is found.
5. Press **OK** to acknowledge the information displayed until the shell prompt appears.
6. Change the file system root as follows:


```
sh-4.2# chroot /mnt/sysimage
```

7. Enter the **passwd** command and follow the instructions displayed on the command line to change the root password.
8. Remove the **autorelabel** file to prevent a time consuming SELinux relabel of the disk:

```
sh-4.2# rm -f /.autorelabel
```

9. Enter the **exit** command to exit the **chroot** environment.
10. Enter the **exit** command again to resume the initialization and finish the system boot.

Procedure 21.3. Resetting the Root Password Using `rd.break`

1. Start the system and, on the GRUB 2 boot screen, press the **e** key for edit.
2. Remove the **rhgb** and **quiet** parameters from the end, or near the end, of the **linux16** line, or **linuxefi** on UEFI systems.

Press **Ctrl+a** and **Ctrl+e** to jump to the start and end of the line, respectively. On some systems, **Home** and **End** might also work.



Important

The **rhgb** and **quiet** parameters must be removed in order to enable system messages.

3. Add the following parameters at the end of the **linux** line on 64-Bit IBM Power Series, the **linux16** line on x86-64 BIOS-based systems, or the **linuxefi** line on UEFI systems:

```
rd.break enforcing=0
```

Adding the **enforcing=0** option enables omitting the time consuming SELinux relabeling process.

The `initramfs` will stop before passing control to the Linux *kernel*, enabling you to work with the root file system.

Note that the `initramfs` prompt will appear on the last console specified on the Linux line.

4. Press **Ctrl+x** to boot the system with the changed parameters.

With an encrypted file system, a password is required at this point. However the password prompt might not appear as it is obscured by logging messages. You can press the **Backspace** key to see the prompt. Release the key and enter the password for the encrypted file system, while ignoring the logging messages.

The `initramfs switch_root` prompt appears.

5. The file system is mounted read-only on **/sysroot/**. You will not be allowed to change the password if the file system is not writable.

Remount the file system as writable:

```
switch_root:/# mount -o remount,rw /sysroot
```

- The file system is remounted with write enabled.

Change the file system's root as follows:

```
switch_root:/# chroot /sysroot
```

The prompt changes to **sh-4.2#**.

- Enter the **passwd** command and follow the instructions displayed on the command line to change the root password.

Note that if the system is not writable, the **passwd** tool fails with the following error:

```
Authentication token manipulation error
```

- Updating the password file results in a file with the incorrect SELinux security context. To relabel all files on next system boot, enter the following command:

```
sh-4.2# touch /.autorelabel
```

Alternatively, to save the time it takes to relabel a large disk, you can omit this step provided you included the **enforcing=0** option in step 3.

- Remount the file system as read only:

```
sh-4.2# mount -o remount,ro /
```

- Enter the **exit** command to exit the **chroot** environment.
- Enter the **exit** command again to resume the initialization and finish the system boot.

With an encrypted file system, a password or phrase is required at this point. However the password prompt might not appear as it is obscured by logging messages. You can press and hold the **Backspace** key to see the prompt. Release the key and enter the password for the encrypted file system, while ignoring the logging messages.



Note

Note that the SELinux relabeling process can take a long time. A system reboot will occur automatically when the process is complete.

- If you added the **enforcing=0** option in step 3 and omitted the **touch /.autorelabel** command in step 8, enter the following command to restore the **/etc/shadow** file's SELinux security context:


```
~]# restorcon /etc/shadow
```

Enter the following commands to turn SELinux policy enforcement back on and verify that it is on:

```
~]# setenforce 1
~]# getenforce
Enforcing
```

21.10. UEFI Secure Boot

The Secure Boot technology ensures that the system firmware checks whether the system boot loader is signed with a cryptographic key authorized by a database contained in the firmware. With signature verification in the next-stage boot loader, kernel, and, potentially, user space, it is possible to prevent the execution of unsigned code.

Secure Boot is the boot path validation component of the Unified Extensible Firmware Interface (UEFI) specification. The specification defines:

- a programming interface for cryptographically protected UEFI variables in non-volatile storage,
- how the trusted X.509 root certificates are stored in UEFI variables,
- validation of UEFI applications like boot loaders and drivers,
- procedures to revoke known-bad certificates and application hashes.

UEFI Secure Boot does not prevent the installation or removal of second-stage boot loaders, nor require explicit user confirmation of such changes. Signatures are verified during booting, not when the boot loader is installed or updated. Therefore, UEFI Secure Boot does not stop boot path manipulations, it simplifies the detection of changes and prevents the system from executing a modified boot path once such a modification has occurred.

21.10.1. UEFI Secure Boot Support in Fedora

Fedora includes support for the UEFI Secure Boot feature, which means that Fedora can be installed and run on systems where UEFI Secure Boot is enabled. On UEFI-based systems with the Secure Boot technology enabled, all drivers that are loaded must be signed with a valid certificate, otherwise the system will not accept them. All drivers provided by Red Hat are signed by the UEFI CA certificate.

If you want to load externally built drivers — drivers that are not provided on the Fedora Linux DVD — you must make sure these drivers are signed as well.

21.11. Additional Resources

Please see the following resources for more information on the GRUB 2 boot loader:

Installed Documentation

- **/usr/share/doc/grub2-tools-`<version-number>`** — This directory contains information about using and configuring GRUB 2. `<version-number>` corresponds to the version of the GRUB 2 package installed.
- **info grub2** — The GRUB 2 info page contains a tutorial, a user reference manual, a programmer reference manual, and a FAQ document about GRUB 2 and its usage.

- **grubby(8)** — The manual page for the command-line tool for configuring GRUB and GRUB 2.
- **new-kernel-pkg(8)** — The manual page for the tool to script kernel installation.

External Documentation

- [Fedora Installation Guide](http://docs.fedoraproject.org/install-guide)¹ — The Installation Guide provides basic information on GRUB 2, for example, installation, terminology, interfaces, and commands.

¹ <http://docs.fedoraproject.org/install-guide>

Manually Upgrading the Kernel

The Fedora kernel is custom-built by the Fedora kernel team to ensure its integrity and compatibility with supported hardware. Before a kernel is released, it must first pass a rigorous set of quality assurance tests.

Fedora kernels are packaged in the RPM format so that they are easy to upgrade and verify using the **DNF** or **PackageKit** package managers. **PackageKit** automatically queries the DNF repositories 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 **dnf**.



Use DNF to install kernels whenever possible

Whenever possible, use either the **DNF** 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.

For more information on installing kernel packages with **DNF**, see [Section 6.1.2, “Updating Packages”](#).

22.1. Overview of Kernel Packages

Fedora 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-headers* — Includes the C header files that specify the interface between the Linux kernel and user-space libraries and programs. The header files define structures and constants that are needed for building most standard programs.
- *linux-firmware* — Contains all of the firmware files that are required by various devices to operate.
- *perf* — This package contains supporting scripts and documentation for the **perf** tool shipped in each kernel image subpackage.
- *kernel-abi-whitelists* — Contains information pertaining to the Fedora kernel ABI, including a lists of kernel symbols that are needed by external Linux kernel modules and a *dnf* plug-in to aid enforcement.
- *kernel-tools* — Contains tools for manipulating the Linux kernel and supporting documentation.

22.2. 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 in case a problem occurs. If the boot loader is not configured properly to boot the new kernel, you can use this media to boot into Fedora.

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, ext4, 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 Fedora installation DVD, or installation CD-ROM#1, and you need a USB storage device formatted with the VFAT file system and around 16 MB of free space. The following procedure will not affect existing files on the USB storage device unless they have the same path names as the files that you copy onto it. To create USB boot media, perform the following commands as the root user:

1. Install the **SYSINUX** bootloader 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
```

5. Copy the **ISOLINUX** files from the **boot.iso** to the USB storage device:

```
~]# cp /mnt/isoboot/isolinux/* /mnt/diskboot
```

6. 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
```

7. Unmount **boot.iso** and the USB storage device:

```
~]# umount /mnt/isoboot /mnt/diskboot
```


8. 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 **dnf 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 might differ:

```
~]# dnf list installed "kernel-*"
Last metadata expiration check performed 0:28:51 ago on Tue May 26 21:22:39 2015.
Installed Packages
kernel-core.x86_64                4.0.3-300.fc22                @System
kernel-core.x86_64                4.0.4-300.fc22                @System
kernel-core.x86_64                4.0.4-301.fc22                @System
kernel-headers.x86_64             4.0.4-301.fc22                @System
kernel-modules.x86_64             4.0.3-300.fc22                @System
kernel-modules.x86_64             4.0.4-300.fc22                @System
kernel-modules.x86_64             4.0.4-301.fc22                @System
```

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 22.1, “Overview of Kernel Packages”](#) for descriptions of the different packages.

22.3. Downloading the Upgraded Kernel

There are several ways to determine if an updated kernel is available for the system.

- Security Advisories — See <http://fedoraproject.org/wiki/FSA> for information on Security Advisories, including kernel upgrades that fix security issues.
- Via Fedora Update System — Download and install the kernel RPM packages. For more information, refer to <http://admin.fedoraproject.org/updates/>.

To install the kernel manually, continue to [Section 22.4, “Performing the Upgrade”](#).

22.4. Performing the Upgrade

After retrieving all of the necessary packages, it is time to upgrade the existing kernel.



Keep the old kernel when performing the upgrade

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 22.5](#), “*Verifying the Initial RAM Disk Image*” for details.

22.5. 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 Fedora 26 systems, whenever a new kernel is installed using either the **DNF**, **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 The Fedora Project.

On architectures that use the GRUB 2 boot loader, you can verify that an *initramfs* corresponding to your current kernel version exists and is specified correctly in the **/boot/grub2/grub.cfg** configuration file by following this procedure:

Procedure 22.1. Verifying the Initial RAM Disk Image

1. As root, list the contents in the **/boot/** directory and find the kernel (**vmlinuz-*kernel_version***) and **initramfs-*kernel_version*** with the latest (most recent) version number:

Example 22.1. Ensuring that the kernel and initramfs versions match

```
~]# ls /boot/
config-3.17.4-302.fc21.x86_64
config-3.17.6-300.fc21.x86_64
config-3.17.7-300.fc21.x86_64
efi
elf-memtest86+-5.01
extlinux
grub2
initramfs-0-rescue-db90b4e3715b42daa871351439343ca4.img
initramfs-3.17.4-302.fc21.x86_64.img
initramfs-3.17.6-300.fc21.x86_64.img
initramfs-3.17.7-300.fc21.x86_64.img
initrd-plymouth.img
lost+found
memtest86+-5.01
System.map-3.17.4-302.fc21.x86_64
System.map-3.17.6-300.fc21.x86_64
System.map-3.17.7-300.fc21.x86_64
vmlinuz-0-rescue-db90b4e3715b42daa871351439343ca4
vmlinuz-3.17.4-302.fc21.x86_64
vmlinuz-3.17.6-300.fc21.x86_64
vmlinuz-3.17.7-300.fc21.x86_64
```

Example 22.1, “Ensuring that the kernel and initramfs versions match” shows that:

- we have three kernels installed (or, more correctly, three kernel files are present in the **/boot/** directory),
- the latest kernel is **vmlinux-3.17.7-300.fc21.x86_64**, and
- an **initramfs** file matching our kernel version, **initramfs-3.17.7-300.fc21.x86_64.img**, also exists.



initrd files in the **/boot/** directory are not the same as **initramfs** files

In the **/boot/** directory you might find several **initrd-kernel_versionkdump.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. For more information on **kdump**, see the [Red Hat Enterprise Linux 7 Kernel Crash Dump Guide](https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Kernel_Crash_Dump_Guide/)¹.

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 might 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
F: Will not override existing initramfs (/boot/initramfs-3.17.7-300.fc21.x86_64.img)
without --force
```

You can create an **initramfs** in the current directory by calling **dracut initramfs_name kernel_version**, for example:

```
~]# dracut "initramfs-$(uname -r).img" $(uname -r)
```

If you need to specify specific kernel modules to be preloaded, add the names of those modules (minus any file name suffixes such as **.ko**) inside the parentheses of the **add_dracutmodules="module [more_modules]"** directive of the **/etc/dracut.conf** configuration file. You can list the file contents of an **initramfs** image file created by **dracut** by using the **lsinitrd initramfs_file** command:

```
~]# lsinitrd /boot/initramfs-3.17.7-300.fc21.x86_64.img
Image: /boot/initramfs-3.17.7-300.fc21.x86_64.img: 18M
=====
Version: dracut-038-31.git20141204.fc21
```

¹ https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Kernel_Crash_Dump_Guide/


```
[output truncated]
```

See `man dracut` and `man dracut.conf` for more information on options and usage.

3. Examine the `/boot/grub2/grub.cfg` configuration file to ensure that an `initramfs-kernel_version.img` file exists for the kernel version you are booting. For example:

```
~]# grep initramfs /boot/grub2/grub.cfg
initrd16 /initramfs-3.17.7-300.fc21.x86_64.img
initrd16 /initramfs-3.17.6-300.fc21.x86_64.img
initrd16 /initramfs-3.17.4-302.fc21.x86_64.img
initrd16 /initramfs-0-rescue-db90b4e3715b42daa871351439343ca4.img
```

See [Section 22.6, “Verifying the Boot Loader”](#) for more information on how to read and update the `/boot/grub2/grub.cfg` file.

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 The Fedora Project; thus, it does not need to be executed manually. To verify that it was created, run the following command as root to make sure the `/boot/vmlinitrd-kernel_version` file already exists:

```
ls -l /boot/
```

The `kernel_version` should match the version of the kernel just installed.

22.6. 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 might not be able to boot into Fedora 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 22.1. Boot loaders by architecture

Architecture	Boot Loader	See
x86	GRUB 2	Section 22.6.1, “Configuring the GRUB 2 Boot Loader”
AMD AMD64 or Intel 64	GRUB 2	Section 22.6.1, “Configuring the GRUB 2 Boot Loader”

Architecture	Boot Loader	See
IBM eServer System i	OS/400	Section 22.6.2, “Configuring the OS/400 Boot Loader”
IBM eServer System p	YABOOT	Section 22.6.3, “Configuring the YABOOT Boot Loader”
IBM System z	z/IPL	—

22.6.1. Configuring the GRUB 2 Boot Loader

Fedora 26 is distributed with GRUB 2, which reads its configuration from the `/boot/grub2/grub.cfg` file. This file is generated by the `grub2-mkconfig` utility based on Linux kernels located in the `/boot` directory, template files located in `/etc/grub.d/`, and custom settings in the `/etc/default/grub` file and is automatically updated each time you install a new kernel from an RPM package. To update this configuration file manually, type the following at a shell prompt as root:

```
grub2-mkconfig -o /boot/grub2/grub.cfg
```

Among various code snippets and directives, the `/boot/grub2/grub.cfg` configuration file contains one or more **menuentry** blocks, each representing a single GRUB 2 boot menu entry. These blocks always start with the **menuentry** keyword followed by a title, list of options, and opening curly bracket, and end with a closing curly bracket. Anything between the opening and closing bracket should be indented. For example, the following is a sample **menuentry** block for Fedora 21 with Linux kernel 3.17.6-300.fc21.x86_64:

```
menuentry 'Fedora (3.17.6-300.fc21.x86_64) 21 (Twenty One)' --class fedora --class gnu-linux
--class gnu --class os --unrestricted $menuentry_id_option 'gnulinux-3.17.4-301.fc21.x86_64-
advanced-efee860-8d55-4e4a-995e-b4c88f9ac9f0' {
    load_video
    set gfxpayload=keep
    insmod gzio
    insmod part_msdos
    insmod ext2
    set root='hd0,msdos1'
    if [ x$feature_platform_search_hint = xy ]; then
        search --no-floppy --fs-uuid --set=root --hint='hd0,msdos1' f19c92f4-9ead-4207-
b46a-723b7a2c51c8
    else
        search --no-floppy --fs-uuid --set=root f19c92f4-9ead-4207-b46a-723b7a2c51c8
    fi
    linux16 /vmlinuz-3.17.6-300.fc21.x86_64 root=/dev/mapper/fedora-root ro
    rd.lvm.lv=fedora/swap rd.lvm.lv=fedora/root rhgb quiet LANG=en_US.UTF-8
    initrd16 /initramfs-3.17.6-300.fc21.x86_64.img
}
```

Each **menuentry** block that represents an installed Linux kernel contains **linux** and **initrd** directives followed by the path to the kernel and the `initramfs` image respectively. If a separate `/boot` partition was created, the paths to the kernel and the `initramfs` image are relative to `/boot`. In the example above, the **initrd16 /initramfs-3.17.6-300.fc21.x86_64.img** line means that the `initramfs` image is actually located at `/boot/initramfs-3.17.6-300.fc21.x86_64.img` when the root file system is mounted, and likewise for the kernel path.

The kernel version number as given on the **linux /vmlinuz-*kernel_version*** line must match the version number of the `initramfs` image given on the **initrd /**

initramfs-kernel_version.img line of each **menuentry** block. For more information on how to verify the initial RAM disk image, refer to [Procedure 22.1, “Verifying the Initial RAM Disk Image”](#).

The **initrd** directive in **grub.cfg** refers to an **initramfs** image

In **menuentry** blocks, the **initrd** directive must point to the location (relative to the **/boot** directory if it is on a separate partition) of the **initramfs** file corresponding to the same kernel version. This directive is called **initrd** because the previous tool which created initial RAM disk images, **mkinitrd**, created what were known as **initrd** files. The **grub.cfg** directive remains **initrd** to maintain compatibility with other tools. The file-naming convention of systems using the **dracut** utility to create the initial RAM disk image is **initramfs-kernel_version.img**.

For information on using **Dracut**, refer to [Section 22.5, “Verifying the Initial RAM Disk Image”](#).

After installing a new kernel with **rpm**, verify that **/boot/grub2/grub.cfg** is correct and reboot the computer into the new kernel. Ensure your hardware is detected by watching the boot process output. If GRUB 2 presents an error and is unable to boot into the new kernel, it is often easiest to try to boot into an alternative or older kernel so that you can fix the problem. Alternatively, use the boot media you created earlier to boot the system.

Causing the GRUB 2 boot menu to display

If you set the **GRUB_TIMEOUT** option in the **/etc/default/grub** file to 0, GRUB 2 will not display its list of bootable kernels when the system starts up. In order to display this list when booting, press and hold any alphanumeric key while and immediately after BIOS information is displayed, and GRUB 2 will present you with the GRUB menu.

22.6.2. 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.

22.6.3. Configuring the YABOOT Boot Loader

IBM eServer System p uses YABOOT as its boot loader. YABOOT uses **/etc/aboot.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 Fedora! 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.

Working with Kernel Modules

The Linux kernel is modular, which means it can extend its capabilities through the use of dynamically-loaded *kernel modules*. A kernel module can provide:

- a device driver which adds support for new hardware; or,
- support for a file system such as `btrfs` or `NFS`.

Like the kernel itself, modules can take parameters that customize their behavior, though the default parameters work well in most cases. User-space tools can list the modules currently loaded into a running kernel; query all available modules for available parameters and module-specific information; and load or unload (remove) modules dynamically into or from a running kernel. Many of these utilities, which are provided by the *kmod* package, take module dependencies into account when performing operations so that manual dependency-tracking is rarely necessary.

On modern systems, kernel modules are automatically loaded by various mechanisms when the conditions call for it. However, there are occasions when it is necessary to load or unload modules manually, such as when one module is preferred over another although either could provide basic functionality, or when a module is misbehaving.

This chapter explains how to:

- use the user-space **kmod** utilities to display, query, load and unload kernel modules and their dependencies;
- set module parameters both dynamically on the command line and permanently so that you can customize the behavior of your kernel modules; and,
- load modules at boot time.

Installing the kmod package

In order to use the kernel module utilities described in this chapter, first ensure the *kmod* package is installed on your system by running, as root:

```
~]# dnf install kmod
```

For more information on installing packages with DNF, see [Section 6.2.4, “Installing Packages”](#).

23.1. Listing Currently-Loaded Modules

You can list all kernel modules that are currently loaded into the kernel by running the **lsmod** command, for example:

```
~]$ lsmod
Module              Size  Used by
tcp_lp              12663  0
bnep                 19704  2
bluetooth           372662  7 bnep
rfkill               26536  3 bluetooth
fuse                 87661  3
```



```

ip6t_rpfilter      12546  1
ip6t_REJECT        12939  2
ipt_REJECT         12541  2
xt_conntrack       12760  7
ebtable_nat        12807  0
ebtable_broute     12731  0
bridge            110196  1 ebtable_broute
stp                12976  1 bridge
llc                14552  2 stp,bridge
ebtable_filter     12827  0
ebtables           30913  3 ebtable_broute,ebtable_nat,ebtable_filter
ip6table_nat       13015  1
nf_conntrack_ipv6  18738  5
nf_defrag_ipv6     34651  1 nf_conntrack_ipv6
nf_nat_ipv6        13279  1 ip6table_nat
ip6table_mangle    12700  1
ip6table_security  12710  1
ip6table_raw       12683  1
ip6table_filter    12815  1
ip6_tables         27025  5
  ip6table_filter,ip6table_mangle,ip6table_security,ip6table_nat,ip6table_raw
iptable_nat        13011  1
nf_conntrack_ipv4  14862  4
nf_defrag_ipv4     12729  1 nf_conntrack_ipv4
nf_nat_ipv4        13263  1 iptable_nat
nf_nat             21798  4 nf_nat_ipv4,nf_nat_ipv6,ip6table_nat,iptable_nat
[output truncated]

```

Each row of **lsmod** output specifies:

- the name of a kernel module currently loaded in memory;
- the amount of memory it uses; and,
- the sum total of processes that are using the module and other modules which depend on it, followed by a list of the names of those modules, if there are any. Using this list, you can first unload all the modules depending the module you want to unload. For more information, see [Section 23.4, “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.

23.2. Displaying Information About a Module

You can display detailed information about a kernel module by running the **modinfo module_name** command.

Module names do not end in .ko

When entering the name of a kernel module as an argument to one of the **kmod** utilities, do not append a **.ko** extension to the end of the name. Kernel module names do not have extensions; their corresponding files do.

Example 23.1. Listing information about a kernel module with **lsmod**

To display information about the **e1000e** module, which is the Intel PRO/1000 network driver, run:


```

~]# modinfo e1000e
filename:      /lib/modules/3.17.4-302.fc21.x86_64/kernel/drivers/net/ethernet/intel/
e1000e/e1000e.ko
version:      2.3.2-k
license:      GPL
description:   Intel(R) PRO/1000 Network Driver
author:       Intel Corporation, <linux.nics@intel.com>
srcversion:   2FBED3F5E2EF40112284D95
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:       ptp
intree:       Y
vermagic:     3.17.4-302.fc21.x86_64 SMP mod_unload
signer:       Fedora kernel signing key
sig_key:      1F:C9:E6:8F:74:19:55:63:48:FD:EE:2F:DE:B7:FF:9D:A6:33:7B:BF
sig_hashalgo: sha256
parm:         debug:Debug level (0=none,...,16=all) (int)
parm:         copybreak:Maximum size of packet that is copied to a new buffer on receive
                (uint)
parm:         TxIntDelay:Transmit Interrupt Delay (array of int)
parm:         TxAbsIntDelay:Transmit Absolute Interrupt Delay (array of int)
parm:         RxIntDelay:Receive Interrupt Delay (array of int)
parm:         RxAbsIntDelay:Receive Absolute Interrupt Delay (array of int)
parm:         InterruptThrottleRate:Interrupt Throttling Rate (array of int)
parm:         IntMode:Interrupt Mode (array of int)
parm:         SmartPowerDownEnable:Enable PHY smart power down (array of int)
parm:         KumeranLockLoss:Enable Kumeran lock loss workaround (array of int)
parm:         WriteProtectNVM:Write-protect NVM [WARNING: disabling this can lead to
                corrupted NVM] (array of int)
parm:         CrcStripping:Enable CRC Stripping, disable if your BMC needs the CRC
                (array of int)

```

Here are descriptions of a few of the fields in **modinfo** output:

filename

The absolute path to the **.ko** kernel object file. You can use **modinfo -n** as a shortcut command for printing only the **filename** field.

description

A short description of the module. You can use **modinfo -d** as a shortcut command for printing only the description field.

alias

The **alias** field appears as many times as there are aliases for a module, or is omitted entirely if there are none.

depends

This field contains a comma-separated list of all the modules this module depends on.

Omitting the depends field

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.

Example 23.2. Listing module parameters

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:

```
~]# modinfo e1000e | grep "^parm" | sort
parm:          copybreak:Maximum size of packet that is copied to a new buffer on
receive (uint)
parm:          CrcStripping:Enable CRC Stripping, disable if your BMC needs the CRC
(array of int)
parm:          debug:Debug level (0=none,...,16=all) (int)
parm:          InterruptThrottleRate:Interrupt Throttling Rate (array of int)
parm:          IntMode:Interrupt Mode (array of int)
parm:          KumeranLockLoss:Enable Kumeran lock loss workaround (array of int)
parm:          RxAbsIntDelay:Receive Absolute Interrupt Delay (array of int)
parm:          RxIntDelay:Receive Interrupt Delay (array of int)
parm:          SmartPowerDownEnable:Enable PHY smart power down (array of int)
parm:          TxAbsIntDelay:Transmit Absolute Interrupt Delay (array of int)
parm:          TxIntDelay:Transmit Interrupt Delay (array of int)
parm:          WriteProtectNVM:Write-protect NVM [WARNING: disabling this can lead to
corrupted NVM] (array of int)
```

23.3. Loading a Module

To load a kernel module, run **modprobe module_name** as root. For example, to load the wacom module, run:

```
~]# modprobe wacom
```

By default, **modprobe** attempts to load the module from **/lib/modules/kernel_version/kernel/drivers/**. In this directory, each type of module has its own subdirectory, such as **net/** and **scsi/**, for network and SCSI interface drivers respectively.

Some modules have dependencies, which are other kernel modules that must be loaded before the module in question can be loaded. The **modprobe** command always takes dependencies into account when performing operations. When you ask **modprobe** to load a specific kernel module, it first examines the dependencies of that module, if there are any, and loads them if they are not already loaded into the kernel. **modprobe** resolves dependencies recursively: it will load all dependencies of dependencies, and so on, if necessary, thus ensuring that all dependencies are always met.

You can use the **-v** (or **--verbose**) option to cause **modprobe** to display detailed information about what it is doing, which can include loading module dependencies.

Example 23.3. modprobe -v shows module dependencies as they are loaded

You can load the Fibre Channel over Ethernet module verbosely by typing the following at a shell prompt:

```
~]# modprobe -v fcoe
insmod /lib/modules/3.17.4-302.fc21.x86_64/kernel/drivers/scsi/scsi_transport_fc.ko.xz
insmod /lib/modules/3.17.4-302.fc21.x86_64/kernel/drivers/scsi/libfc/libfc.ko.xz
insmod /lib/modules/3.17.4-302.fc21.x86_64/kernel/drivers/scsi/fcoe/libfcoe.ko.xz
insmod /lib/modules/3.17.4-302.fc21.x86_64/kernel/drivers/scsi/fcoe/fcoe.ko.xz
```

In this example, you can see 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.



Always use modprobe instead of insmod!

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.

23.4. 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 23.1, “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.

Example 23.4. Unloading a kernel module

For example, if you want to unload the `firewire_ohci` module, your terminal session might look similar to this:

```
~]# modinfo -F depends firewire_ohci
firewire-core
~]# modinfo -F depends firewire_core
crc-itu-t
~]# modinfo -F depends crc-itu-t
```


You have figured out the dependency tree (which does not branch in this example) for the loaded Firewire modules: `firewire_ohci` depends on `firewire_core`, which itself depends on `crc_itu_t`.

You can unload `firewire_ohci` using the `modprobe -v -r module_name` command, where `-r` is short for `--remove` and `-v` for `--verbose`:

```
~]# modprobe -r -v firewire_ohci
rmmod firewire_ohci
rmmod firewire_core
rmmod crc_itu_t
```

The output shows that modules are unloaded in the reverse order that they are loaded, given that no processes depend on any of the modules being unloaded.



Do not use `rmmod` directly!

Although the `rmmod` command can be used to unload kernel modules, it is recommended to use `modprobe -r` instead.

23.5. Setting Module Parameters

Like the kernel itself, modules can also take parameters that change their behavior. Most of the time, the default ones work well, but occasionally it is necessary or desirable to set custom parameters for a module. Because parameters cannot be dynamically set for a module that is already loaded into a running kernel, there are two different methods for setting them.

1. You can unload all dependencies of the module you want to set parameters for, unload the module using `modprobe -r`, and then load it with `modprobe` along with a list of customized parameters. This method is often used when the module does not have many dependencies, or to test different combinations of parameters without making them persistent, and is the method covered in this section.
2. Alternatively, you can list the new parameters in an existing or newly created file in the `/etc/modprobe.d/` directory. This method makes the module parameters persistent by ensuring that they are set each time the module is loaded, such as after every reboot or `modprobe` command. This method is covered in [Section 23.6, “Persistent Module Loading”](#), though the following information is a prerequisite.

Example 23.5. Supplying optional parameters when loading a kernel module

You can use `modprobe` to load a kernel module with custom parameters using the following command line format:

```
~]# modprobe module_name [parameter=value]
```

When loading a module with custom parameters on the command line, be aware of the following:

- You can enter multiple parameters and values by separating them with spaces.

- Some module parameters expect a list of comma-separated values as their argument. When entering the list of values, do *not* insert a space after each comma, or **modprobe** will incorrectly interpret the values following spaces as additional parameters.
- The **modprobe** command silently succeeds with an exit status of 0 if:
 - it successfully loads the module, or
 - the module is *already* loaded into the kernel.

Thus, you must ensure that the module is not already loaded before attempting to load it with custom parameters. The **modprobe** command does not automatically reload the module, or alert you that it is already loaded.

Here are the recommended steps for setting custom parameters and then loading a kernel module. This procedure illustrates the steps using the `e1000e` module, which is the network driver for Intel PRO/1000 network adapters, as an example:

Procedure 23.1. Loading a Kernel Module with Custom Parameters

1. First, ensure the module is not already loaded into the kernel:

```
~]# lsmod |grep e1000e
~]#
```

Output would indicate that the module is already loaded into the kernel, in which case you must first unload it before proceeding. See [Section 23.4, “Unloading a Module”](#) for instructions on safely unloading it.

2. Load the module and list all custom parameters after the module name. For example, if you wanted to load the Intel PRO/1000 network driver with the interrupt throttle rate set to 3000 interrupts per second for the first, second, and third instances of the driver, and turn on debug, you would run, as root:

```
~]# modprobe e1000e InterruptThrottleRate=3000,3000,3000 debug=1
```

This example illustrates passing multiple values to a single parameter by separating them with commas and omitting any spaces between them.

23.6. Persistent Module Loading

As shown in [Example 23.1, “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 the `systemd-modules-load.service` daemon by creating a **program.conf** file in the `/etc/modules-load.d/` directory, where *program* is any descriptive name of your choice. The files in `/etc/modules-load.d/` are text files that list the modules to be loaded, one per line.

Example 23.6. A Text File to Load a Module

To create a file to load the **virtio-net.ko** module, create a file `/etc/modules-load.d/virtio-net.conf` with the following content:

```
# Load virtio-net.ko at boot
virtio-net
```


See the `modules-load.d(5)` and `systemd-modules-load.service(8)` man pages for more information.

23.7. Signing Kernel Modules for Secure Boot

Fedora includes support for the UEFI Secure Boot feature, which means that Fedora can be installed and run on systems where UEFI Secure Boot is enabled.¹ When Secure Boot is enabled, the EFI operating system boot loaders, the Fedora kernel, and all kernel modules must be signed with a private key and authenticated with the corresponding public key. The Fedora distribution includes signed boot loaders, signed kernels, and signed kernel modules. In addition, the signed first-stage boot loader and the signed kernel include embedded Fedora public keys. These signed executable binaries and embedded keys enable Fedora to install, boot, and run with the Microsoft UEFI Secure Boot CA keys that are provided by the UEFI firmware on systems that support UEFI Secure Boot.²

The information provided in the following sections describes steps necessary to enable you to self-sign privately built kernel modules for use with Fedora on UEFI-based systems where Secure Boot is enabled. These sections also provide an overview of available options for getting your public key onto the target system where you want to deploy your kernel module.

23.7.1. Prerequisites

In order to enable signing of externally built modules, the tools listed in the following table are required to be installed on the system.

Table 23.1. Required Tools

Tool	Provided by Package	Used on	Purpose
openssl	<i>openssl</i>	Build system	Generates public and private X.509 key pair
sign-file	<i>kernel-devel</i>	Build system	Perl script used to sign kernel modules
perl	<i>perl</i>	Build system	Perl interpreter used to run the signing script
mokutil	<i>mokutil</i>	Target system	Optional tool used to manually enroll the public key
keyctl	<i>keyutils</i>	Target system	Optional tool used to display public keys in the system key ring

Note

Note that the build system, where you build and sign your kernel module, does not need to have UEFI Secure Boot enabled and does not even need to be a UEFI-based system.

¹ Fedora does not require the use of Secure Boot on UEFI systems.

² Not all UEFI-based systems include support for Secure Boot.

23.7.2. Kernel Module Authentication

In Fedora, when a kernel module is loaded, the module's signature is checked using the public X.509 keys on the kernel's system key ring, excluding those keys that are on the kernel's system black list key ring.

23.7.2.1. Sources For Public Keys Used To Authenticate Kernel Modules

During boot, the kernel loads X.509 keys into the system key ring or the system black list key ring from a set of persistent key stores as shown in [Table 23.2, “Sources For System Key Rings”](#)

Table 23.2. Sources For System Key Rings

Source of X.509 Keys	User Ability to Add Keys	UEFI Secure Boot State	Keys Loaded During Boot
Embedded in kernel	No	-	.system_keyring
UEFI Secure Boot "db"	Limited	Not enabled	No
		Enabled	.system_keyring
UEFI Secure Boot "dbx"	Limited	Not enabled	No
		Enabled	.system_keyring
Embedded in shim.efi boot loader	No	Not enabled	No
		Enabled	.system_keyring
Machine Owner Key (MOK) list	Yes	Not enabled	No
		Enabled	.system_keyring

Note that if the system is not UEFI-based or if UEFI Secure Boot is not enabled, then only the keys that are embedded in the kernel are loaded onto the system key ring and you have no ability to augment that set of keys without rebuilding the kernel. The system black list key ring is a list of X.509 keys which have been revoked. If your module is signed by a key on the black list then it will fail authentication even if your public key is in the system key ring.

To confirm if Secure Boot is enabled, enter a command as follows:

```
~]$ mokutil --sb-state
SecureBoot enabled
```

If Secure Boot is not enabled then the message **Failed to read SecureBoot** is displayed.

You can display information about the keys on the system key rings using the **keyctl** utility. The following is abbreviated example output from a Fedora system where UEFI Secure Boot is not enabled.

```
~]# keyctl list %:.system_keyring
1 key in keyring:
265061799: ---lswrv 0 0 asymmetric: Fedora kernel signing key:
ba8e2919f98f3f8e2e27541cde0d1f...
```

The following is abbreviated example output from a Fedora system where UEFI Secure Boot is enabled.

```
~]# keyctl list %:.system_keyring
5 keys in keyring:
...asymmetric: Microsoft Windows Production PCA 2011: a92902398e16c497...
...asymmetric: Fedora kernel signing key: ba8e2919f98f3f8e2e27541cde0d...
...asymmetric: Fedora Secure Boot CA: fde32599c2d61db1bf5807335d7b20e4...
```



```
...asymmetric: Red Hat Test Certifying CA: 08a0ef5800cb02fb587c12b4032...
...asymmetric: Microsoft Corporation UEFI CA 2011: 13adbf4309bd82709c8...
```

The above output shows the addition of two keys from the UEFI Secure Boot "db" keys plus the **Fedora Secure Boot CA** which is embedded in the `shim.efi` boot loader.

23.7.2.2. Kernel Module Authentication Requirements

If UEFI Secure Boot is enabled or if the `module.sig_enforce` kernel parameter has been specified, then only signed kernel modules that are authenticated using a key on the system key ring can be successfully loaded.³ If UEFI Secure Boot is disabled and if the `module.sig_enforce` kernel parameter has not been specified, then unsigned kernel modules and signed kernel modules without a public key can be successfully loaded. This is summarized in [Table 23.3, “Kernel Module Authentication Requirements for Loading”](#).

Table 23.3. Kernel Module Authentication Requirements for Loading

Module Signed	Public Key Found and Signature Valid	UEFI Secure Boot State	module.sig_enf	Module Load	Kernel Tainted
Unsigned	-	Not enabled	Not enabled	Succeeds	Yes
		Not enabled	Enabled	Fails	
		Enabled	-	Fails	-
Signed	No	Not enabled	Not enabled	Succeeds	Yes
		Not enabled	Enabled	Fails	-
		Enabled	-	Fails	-
Signed	Yes	Not enabled	Not enabled	Succeeds	No
		Not enabled	Enabled	Succeeds	No
		Enabled	-	Succeeds	No

Subsequent sections will describe how to generate a public and private X.509 key pair, how to use the private key to sign a kernel module, and how to enroll the public key into a source for the system key ring.

23.7.3. Generating a Public and Private X.509 Key Pair

You need to generate a public and private X.509 key pair that will be used to sign a kernel module after it has been built. The corresponding public key will be used to authenticate the kernel module when it is loaded.

1. The **openssl** tool can be used to generate a key pair that satisfies the requirements for kernel module signing in Fedora. Some of the parameters for this key generation request are best specified with a configuration file; follow the example below to create your own configuration file.

```
~]# cat << EOF > configuration_file.config
[ req ]
default_bits = 4096
distinguished_name = req_distinguished_name
prompt = no
string_mask = utf8only
```

³ Provided that the public key is not on the system black list key ring.


```
x509_extensions = myexts

[ req_distinguished_name ]
0 = Organization
CN = Organization signing key
emailAddress = E-mail address

[ myexts ]
basicConstraints=critical,CA:FALSE
keyUsage=digitalSignature
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid
EOF
```

2. After you have created the configuration file, you can create an X.509 public and private key pair. The public key will be written to the **public_key.der** file and the private key will be written to the **private_key.priv** file.

```
~]# openssl req -x509 -new -nodes -utf8 -sha256 -days 36500 \> -batch -
config configuration_file.config -outform DER \> -out public_key.der \> -
keyout private_key.priv
```

3. Enroll your public key on all systems where you want to authenticate and load your kernel module.



Warning

Take proper care to guard the contents of your private key. In the wrong hands, the key could be used to compromise any system which has your public key.

23.7.4. Enrolling Public Key on Target System

When Fedora boots on a UEFI-based system with Secure Boot enabled, all keys that are in the Secure Boot db key database, but not in the dbx database of revoked keys, are loaded onto the system keyring by the kernel. The system keyring is used to authenticate kernel modules.

23.7.4.1. Factory Firmware Image Including Public Key

To facilitate authentication of your kernel module on your systems, consider requesting your system vendor to incorporate your public key into the UEFI Secure Boot key database in their factory firmware image.

23.7.4.2. Executable Key Enrollment Image Adding Public Key

It is possible to add a key to an existing populated and active Secure Boot key database. This can be done by writing and providing an EFI executable *enrollment* image. Such an enrollment image contains a properly formed request to append a key to the Secure Boot key database. This request must include data that is properly signed by the private key that corresponds to a public key that is already in the system's Secure Boot Key Exchange Key (KEK) database. Additionally, this EFI image must be signed by a private key that corresponds to a public key that is already in the key database.

It is also possible to write an enrollment image that runs under Fedora. However, the Fedora image must be properly signed by a private key that corresponds to a public key that is already in the KEK database.

The construction of either type of key enrollment images requires assistance from the platform vendor.

23.7.4.3. System Administrator Manually Adding Public Key to the MOK List

The Machine Owner Key (MOK) facility is a feature that is supported by Fedora and can be used to augment the UEFI Secure Boot key database. When Fedora boots on a UEFI-enabled system with Secure Boot enabled, the keys on the MOK list are also added to the system keyring in addition to the keys from the key database. The MOK list keys are also stored persistently and securely in the same fashion as the Secure Boot key database keys, but these are two separate facilities. The MOK facility is supported by `shim.efi`, `MokManager.efi`, `grubx64.efi`, and the Fedora **`mokutil`** utility.

The major capability provided by the MOK facility is the ability to add public keys to the MOK list without needing to have the key chain back to another key that is already in the KEK database. However, enrolling a MOK key requires manual interaction by a *physically present* user at the UEFI system console on each target system. Nevertheless, the MOK facility provides an excellent method for testing newly generated key pairs and testing kernel modules signed with them.

Follow these steps to add your public key to the MOK list:

1. Request addition of your public key to the MOK list using a Fedora userspace utility:

```
~]# mokutil --import my_signing_key_pub.der
```

You will be asked to enter and confirm a password for this MOK enrollment request.

2. Reboot the machine.
3. The pending MOK key enrollment request will be noticed by `shim.efi` and it will launch `MokManager.efi` to allow you to complete the enrollment from the UEFI console. You will need to enter the password you previously associated with this request and confirm the enrollment. Your public key is added to the MOK list, which is persistent.

Once a key is on the MOK list, it will be automatically propagated to the system key ring on this and subsequent boots when UEFI Secure Boot is enabled.

23.7.5. Signing Kernel Module with the Private Key

There are no extra steps required to prepare your kernel module for signing. You build your kernel module normally. Assuming an appropriate Makefile and corresponding sources, follow these steps to build your module and sign it:

1. Build your **`my_module.ko`** module the standard way:

```
~]# make -C /usr/src/kernels/$(uname -r) M=$PWD modules
```

2. Sign your kernel module with your private key. This is done with a Perl script. Note that the script requires that you provide both the files that contain your private and the public key as well as the kernel module file that you want to sign.

```
~]# perl /usr/src/kernels/$(uname -r)/scripts/sign-file \> sha256 \>  
my_signing_key.priv \> my_signing_key_pub.der \> my_module.ko
```

Your kernel module is in ELF image format and this script computes and appends the signature directly to the ELF image in your **`my_module.ko`** file. The **`modinfo`** utility can be used to display

information about the kernel module's signature, if it is present. For information on using the utility, see [Section 23.2, “Displaying Information About a Module”](#).

Note that this appended signature is not contained in an ELF image section and is not a formal part of the ELF image. Therefore, tools such as **readelf** will not be able to display the signature on your kernel module.

Your kernel module is now ready for loading. Note that your signed kernel module is also loadable on systems where UEFI Secure Boot is disabled or on a non-UEFI system. That means you do not need to provide both a signed and unsigned version of your kernel module.

23.7.6. Loading Signed Kernel Module

Once your public key is enrolled and is in the system keyring, the normal kernel module loading mechanisms will work transparently. In the following example, you will use **mokutil** to add your public key to the MOK list and you will manually load your kernel module with **modprobe**.

1. Optionally, you can verify that your kernel module will not load before you have enrolled your public key. First, verify what keys have been added to the system key ring on the current boot by running the **keyctl list %:.system_keyring** as root. Since your public key has not been enrolled yet, it should not be displayed in the output of the command.

2. Request enrollment of your public key.

```
~]# mokutil --import my_signing_key_pub.der
```

3. Reboot, and complete the enrollment at the UEFI console.

```
~]# reboot
```

4. After the system reboots, verify the keys on the system key ring again.

```
~]# keyctl list %:.system_keyring
```

5. You should now be able to load your kernel module successfully.

```
~]# modprobe -v my_module
insmod /lib/modules/3.17.4-302.fc21.x86_64/extra/my_module.ko
~]# lsmod | grep my_module
my_module 12425 0
```

23.8. Additional Resources

For more information on kernel modules and their utilities, see the following resources.

Manual Page 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 and External Documentation

- [Linux Loadable Kernel Module HOWTO](http://tldp.org/HOWTO/Module-HOWTO/)⁴ — The *Linux Loadable Kernel Module HOWTO* from the Linux Documentation Project contains further information on working with kernel modules.

⁴ <http://tldp.org/HOWTO/Module-HOWTO/>

Appendix A. RPM

The *RPM Package Manager* (**RPM**) is an open packaging system that runs on Fedora as well as other Linux and UNIX systems. Red Hat and the Fedora Project encourage other vendors to use **RPM** for their own products. **RPM** is distributed under the terms of the *GPL* (*GNU General Public License*).

The **RPM Package Manager** only works with packages built in the *RPM format*. **RPM** itself is provided as the pre-installed *rpm* package. For the end user, **RPM** makes system updates easy. Installing, uninstalling, and upgrading **RPM** packages can be accomplished with short commands. **RPM** maintains a database of installed packages and their files, so you can make queries and verify installed files on your system. There are several applications, such as **DNF** or **PackageKit**, that can make working with packages in the **RPM** format even easier.



Use DNF Instead of RPM Whenever Possible

For most package-management tasks, the **DNF** package manager offers equal and often greater capabilities and utility than **RPM**. **DNF** also performs and tracks complicated system-dependency resolutions. **DNF** maintains the system integrity and forces a system integrity check if packages are installed or removed using another application, such as **RPM**, instead of **DNF**. For these reasons, it is highly recommended that you use **DNF** instead of **RPM** whenever possible to perform package-management tasks. See [Chapter 6, DNF](#).

If you prefer a graphical interface, you can use the **PackageKit** GUI application, which uses **DNF** as its back end, to manage your system's packages.

During upgrades, **RPM** handles configuration files carefully, so that you never lose your customizations — something that you cannot accomplish with regular `.tar.gz` files.

For the developer, **RPM** enables software source code to be packaged into source and binary packages for end users. This process is quite simple and is driven from a single file and optional patches that you create. This clear delineation between pristine sources and your patches along with build instructions eases the maintenance of the package as new versions of the software are released.



Note

Because **RPM** can make changes to the system itself, performing operations like installing, upgrading, downgrading, and uninstalling binary packages system-wide requires root privileges in most cases.

A.1. RPM Design Goals

To understand how to use **RPM**, it is helpful to understand the design goals of **RPM**:

Upgradability

With **RPM**, you can upgrade individual components of your system without a complete reinstallation. When you get a new release of an operating system based on **RPM**, such as

Fedora, you do not need to reinstall a fresh copy of the operating system on your machine (as you might need to with operating systems based on other packaging systems). **RPM** allows for intelligent, fully-automated, in-place upgrades of your system. In addition, configuration files in packages are preserved across upgrades, so you do not lose your customizations. There are no special upgrade files needed to upgrade a package because the same **RPM** file is used to both install and upgrade the package on the system.

Powerful Querying

RPM is designed to provide powerful querying options. You can perform searches on your copy of the database for packages or even just certain files. You can also easily find out what package a file belongs to and where the package came from. The files an **RPM** package contains are in a compressed archive, with a custom binary header containing useful information about the package and its contents, allowing you to query individual packages quickly and easily.

System Verification

Another powerful **RPM** feature is the ability to verify packages. It allows you to verify that the files installed on the system are the same as the ones supplied by a given package. If an inconsistency is detected, **RPM** notifies you, and you can reinstall the package if necessary. Any configuration files that you modified are preserved during reinstallation.

Pristine Sources

A crucial design goal 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.

A.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 see `rpm(8)`. Also, see [Section A.5, “Additional Resources”](#) for more information on **RPM**.

A.2.1. Installing and Upgrading Packages

RPM packages typically have file names in the following form:

```
package_name-version-release-operating_system-CPU_architecture.rpm
```

For example the `tree-1.7.0-3.fc26.x86_64.rpm` file name includes the package name (**tree**), version (**1.7.0**), release (**3**), operating system major version (**fc26**) and CPU architecture (**x86_64**).



Important

When installing a package, ensure it is compatible with your operating system and processor architecture. This can usually be determined by checking the package name. For example, the file name of an **RPM** package compiled for the AMD64/Intel 64 computer architectures ends with **x86_64.rpm**.

The **-U** (or **--upgrade**) option has two functions, it can be used to:

- upgrade an existing package on the system to a newer version, or
- install a package if an older version is not already installed.

The **rpm -U package.rpm** command is therefore able to either *upgrade* or *install*, depending on the presence of an older version of *package.rpm* on the system.

Assuming the **tree-1.7.0-3.fc26.x86_64.rpm** package is in the current directory, log in as root and type the following command at a shell prompt to either upgrade or install the *tree* package:

```
~]# rpm -Uvh tree-1.7.0-3.fc26.x86_64.rpm
```



Use -Uvh for nicely-formatted RPM installs

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 or installation is successful, the following output is displayed:

```
Preparing... ##### [100%]
 1:tree      ##### [100%]
```




Always use the **-i** (install) option to install new kernel packages!

rpm provides two different options for installing packages: the aforementioned **-U** option (which historically stands for *upgrade*), and the **-i** option (which historically stands for *install*). Because the **-U** option includes both install and upgrade functions, the use of **rpm -Uvh** with all packages, *except kernel packages*, is recommended.

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 22, Manually Upgrading the Kernel](#).

The signature of a package is checked automatically when installing or upgrading a package. The signature confirms that the package was signed by an authorized party. If the verification of the signature fails, an error message is displayed.

If you do not have the appropriate key installed to verify the signature, the message contains the word **NOKEY**:

```
warning: tree-1.7.0-3.fc26.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID 431d51: NOKEY
```

See [Section A.3.2, “Checking Package Signatures”](#) for more information on checking package signatures.

A.2.1.1. Replacing Already-Installed Packages

If a package of the same name and version is already installed, the following output is displayed:

```
Preparing... ##### [100%]
package tree-1.7.0-3.fc26.x86_64 is already installed
```

To install the package anyway, use the **--replacepks** option, which tells **RPM** to ignore the error:

```
~]# rpm -Uvh --replacepks tree-1.7.0-3.fc26.x86_64.rpm
```

This option is helpful if files installed from the package were deleted or if you want the original configuration files to be installed.

If you attempt an upgrade to an *older* version of a package (that is, if a newer version of the package is already installed), **RPM** informs you that a newer version is already installed. To force **RPM** to perform the downgrade, use the **--oldpackage** option:

```
rpm -Uvh --oldpackage older_package.rpm
```

A.2.1.2. Resolving File Conflicts

If you attempt to install a package that contains a file that has already been installed by another package, a conflict message is displayed. To make **RPM** ignore this error, use the **--replacefiles** option:

```
rpm -Uvh --replacefiles package.rpm
```

A.2.1.3. Satisfying Unresolved Dependencies

RPM packages sometimes depend on other packages, which means that they require other packages to be installed to run properly. If you try to install a package that has an unresolved dependency, a message about a failed dependency is displayed.

Find the suggested package(s) on the Fedora installation media or on one of the active Fedora mirrors and add it to the installation command. To determine which package contains the required file, use the **--whatprovides** option:

```
rpm -q --whatprovides "required_file"
```

If the package that contains *required_file* is in the **RPM** database, the name of the package is displayed.



Warning

Although you can *force* **rpm** to install a package that has an unresolved dependency (using the **--nodeps** option), this is *not* recommended and will usually result in the installed software failing to run. Installing packages with **--nodeps** can cause applications to misbehave or terminate unexpectedly. It can also cause serious package-management problems or system failure. For these reasons, heed the warnings about missing dependencies. The **DNF** package manager performs automatic dependency resolution and fetches dependencies from on-line repositories.

A.2.1.4. Preserving Changes in Configuration Files

Because **RPM** performs intelligent upgrading of packages with configuration files, you may see the following message:

```
saving /etc/configuration_file.conf as /etc/configuration_file.conf.rpmnew
```

This message means that the changes you made to the configuration file may not be *forward-compatible* with the new configuration file in the package, so **RPM** saved your original file and installed a new one. You should investigate the differences between the two configuration files and resolve them as soon as possible to ensure that your system continues to function properly.

Alternatively, **RPM** may save the package's *new* configuration file as, for example, **configuration_file.conf.rpmnew** and leave the configuration file you modified untouched. You should still resolve any conflicts between your modified configuration file and the new one, usually by merging changes from the old one to the new one, for example using the **diff** program.

A.2.2. Uninstalling Packages

Uninstalling a package is just as simple as installing one. Type the following command at a shell prompt as root:

```
rpm -e package
```

rpm -e and package name errors

Note that the command expects only the package *name*, not the name of the original package *file*. If you attempt to uninstall a package using the **rpm -e** command and provide the original full file name, you receive a package-name error.

You can encounter dependency errors when uninstalling a package if another installed package depends on the one you are trying to remove. For example:

```
~]# rpm -e ghostscript
error: Failed dependencies:
  ghostscript is needed by (installed) ghostscript-cups-9.07-16.fc26.x86_64
  ghostscript is needed by (installed) foomatic-4.0.9-6.fc26.x86_64
  libgs.so.9()(64bit) is needed by (installed) libspectre-0.2.7-4.fc26.x86_64
  libijs-0.35.so()(64bit) is needed by (installed) gutenprint-5.2.9-15.fc26.x86_64
  libijs-0.35.so()(64bit) is needed by (installed) cups-filters-1.0.35-15.fc26.x86_64
```



Warning: Forcing Package Installation

Although you can *force* **rpm** to uninstall a package that has unresolved dependencies (using the **- -nodeps** option), this is *not* recommended. Removing packages with **- -nodeps** can cause applications from the packages whose dependencies are removed to misbehave or terminate unexpectedly. It can also cause serious package-management problems or system failure. For these reasons, heed the warnings about failed dependencies.

A.2.3. Freshening Packages

Freshening is similar to upgrading, except that only installed packages are upgraded. Type the following command at a shell prompt as root:

```
rpm -Fvh package.rpm
```

The **-F** (or **- -freshen**) option compares the versions of the packages specified on the command line with the versions of packages that are already installed on the system. When a newer version of an already-installed package is processed by the **- -freshen** option, it is upgraded to the newer version. However, the **- -freshen** option does not install a package if no previously-installed package of the same name exists. This differs from regular upgrading, as an upgrade installs all specified packages regardless of whether or not older versions of the packages are already installed.

Freshening works for single packages or package groups. For example, freshening can help if you download a large number of different packages, and you only want to upgrade those packages that

are already installed on the system. In this case, issue the following command with the ***.rpm** global expression:

```
~]# rpm -Fvh *.rpm
```

RPM then automatically upgrades only those packages that are already installed.

A.2.4. Querying Packages

The **RPM** database stores information about all **RPM** packages installed on the system. It is stored in the **/var/lib/rpm/** directory and is used for many things, including querying what packages are installed, what version each package is, and for calculating changes to files in packages since their installation. To query this database, use the **rpm** command with the **-q** (or **--query**) option:

```
rpm -q package_name
```

This command displays the package name, version, and release number of the installed package *package_name*. For example:

```
~]$ rpm -q tree
tree-1.7.0-3.fc26.x86_64
```

See the **Package Selection Options** subheading in the **rpm(8)** manual page for a list of options that can be used to further refine or qualify your query. Use options listed below the **Package Query Options** subheading to specify what information to display about the queried packages.

A.2.5. Verifying Packages

Verifying a package is comparing information about files on the system installed from a package with the same information from the original package. Among other parameters, verifying compares the file size, MD5 sum, permissions, type, owner, and the group of each file.

Use the **rpm** command with the **-V** (or **--verify**) option to verify packages. For example:

```
~]$ rpm -V tree
```

See the **Package Selection Options** subheading in the **rpm(8)** manual page for a list of options that can be used to further refine or qualify your query. Use options listed below the **Verify Options** subheading to specify what characteristics to verify in the queried packages.

If everything verifies properly, there is no output. If there are any discrepancies, they are displayed. The output consists of lines similar to these:

```
~]# rpm -V abrt
S.5....T. c /etc/abrt/abrt.conf
.M..... /var/spool/abrt-upload
```

The format of the output is a string of nine characters followed by an optional attribute marker and the name of the processed file.

The first nine characters are the results of tests performed on the file. Each test is the comparison of one attribute of the file to the value of that attribute as recorded in the **RPM** database. A single period

(.) means the test passed, and the question-mark character (?) signifies that the test could not be performed. The following table lists symbols that denote specific discrepancies:

Table A.1. RPM Verification Symbols

Symbol	Description
S	file size differs
M	mode differs (includes permissions and file type)
5	digest (formerly MD5 sum) differs
D	device major/minor number mismatch
L	readLink(2) path mismatch
U	user ownership differs
G	group ownership differs
T	mtime differs
P	capabilities differ

The attribute marker, if present, describes the purpose of the given file. The following table lists the available attribute markers:

Table A.2. RPM Verification Symbols

Marker	Description
c	configuration file
d	documentation file
l	license file
r	readme file

If you see any output, use your best judgment to determine if you should remove the package, reinstall it, or fix the problem in another way.

A.3. Finding and Verifying RPM Packages

Before using any **RPM** packages, you must know where to find them and be able to verify if you can trust them.

A.3.1. Finding RPM Packages

Although there are many **RPM** repositories on the Internet, for security and compatibility reasons, you should consider installing only official Fedora-provided RPM packages. The following is a list of sources for **RPM** packages:

- Official Fedora installation media.
- Official **RPM** repositories provided with the **DNF** package manager. See [Chapter 6, DNF](#) for details on how to use the official Fedora package repositories.
- Unofficial, third-party repositories not affiliated with The Fedora Project also provide RPM packages.

**Important**

When considering third-party repositories for use with your Fedora 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 Fedora repositories.

A.3.2. Checking Package Signatures

RPM packages can be signed using **GNU Privacy Guard** (or **GPG**), which helps you make certain that downloaded packages are trustworthy. **GPG** is a tool for secure communication. With **GPG**, you can authenticate the validity of documents and encrypt or decrypt data.

To verify that a package has not been corrupted or tampered with, check its **GPG** signature by using the **rpmkeys** command with the **-K** (or **--checksig**) option:

```
rpmkeys -K package.rpm
```

Note that the **DNF** package manager performs automatic checking of **GPG** signatures during installations and upgrades.

GPG is installed by default, as well as a set of Red Hat keys for verifying packages. To import additional keys for use with **RPM**, see [Section A.3.2.1, “Importing GPG Keys”](#).

A.3.2.1. Importing GPG Keys

To verify Red Hat packages, a Red Hat **GPG** key needs to be installed. A set of basic keys is installed by default. To view a list of installed keys, execute the following command at a shell prompt:

```
~]$ rpm -qa gpg-pubkey*
```

To display details about a specific key, use **rpm -qi** followed by the output from the previous command. For example:

```
~]$ rpm -qi gpg-pubkey-fd431d51-4ae0493b
```

Use the **rpmkeys** command with the **--import** option to install a new key for use with **RPM**. The default location for storing **RPM** **GPG** keys is the **/etc/pki/rpm-gpg/** directory. To import new keys, use a command like the following as root:

```
~)# rpmkeys --import /etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
```

See the [Product Signing \(GPG\) Keys](#)¹ article on the Red Hat Customer Portal for additional information about Red Hat package-signing practices.

¹ <https://access.redhat.com/security/team/key/>

A.4. Common Examples of RPM Usage

RPM is a useful tool for both managing your system and diagnosing and fixing problems. See the following examples for an overview of some of the most-used options.

- To verify your entire system and see what files are missing, issue the following command as root:

```
rpm -Va
```

If some files are missing or appear corrupted, consider reinstalling relevant packages.

- To determine which package owns a file, enter:

```
rpm -qf file
```

- To verify the package that owns a particular file, enter as root:

```
rpm -vf file
```

- To locate documentation files that are a part of a package to which a file belongs, enter:

```
rpm -qdf file
```

- To find information about a (non-installed) package file, use the following command:

```
rpm -qip package.rpm
```

- To list files contained in a package, use:

```
rpm -qlp package.rpm
```

See the rpm(8) manual page for more options.

A.5. Additional Resources

RPM is a complex utility with many options and methods for querying, installing, upgrading, and removing packages. See the following resources to learn more about **RPM**.

Installed Documentation

- **rpm --help** — This command displays a quick reference of **RPM** parameters.

- `rpm(8)` — The **RPM** manual page offers an overview of all available **RPM** parameters.

Online Documentation

- The **RPM** website — <http://www.rpm.org/>
- The **RPM** mailing list — <http://lists.rpm.org/mailman/listinfo/rpm-list>

See Also

- [Chapter 6, DNF](#) describes how to use the **DNF** package manager to search, install, update, and uninstall packages on the command line.

Appendix B. Revision History

Revision 1-9 **Tue Jul 11 2017** **Petr Bokoč** pbokoc@redhat.com

Fedora 26 release of the *System Administrator's Guide*.

Revision 1-8 **Mon Nov 14 2016** **Petr Bokoč** pbokoc@redhat.com

Fedora 25 release of the *System Administrator's Guide*.

Revision 1-7 **Tue June 21 2016** **Stephen Wadeley** swadeley@redhat.com

Fedora 24 release of the *System Administrator's Guide*.

Revision 1-5 **Mon Nov 02 2015** **Stephen Wadeley** swadeley@redhat.com

Fedora 23 release of the *System Administrator's Guide*.

Revision 1-4.1 **Tue Oct 27 2015** **Stephen Wadeley** swadeley@redhat.com

Added "Gaining Privileges" chapter, "Using OpenSSH Certificate Authentication" section, and made improvements to the GRUB 2 chapter.

Revision 1-4 **Mon May 25 2015** **Stephen Wadeley** swadeley@redhat.com

Fedora 22 release of the *System Administrator's Guide*.

Revision 1-3 **Mon Apr 4 2015** **Stephen Wadeley** swadeley@redhat.com

Replaced Yum chapter with DNF chapter.

Revision 1-2.1 **Wed Mar 4 2015** **Stephen Wadeley** swadeley@redhat.com

Added "Working with the GRUB 2 Boot Loader" chapter.

Revision 1-2 **Tue Dec 9 2014** **Stephen Wadeley** swadeley@redhat.com

Fedora 21 release of the *System Administrator's Guide*.

Revision 1-1 **Thu Aug 9 2012** **Jaromír Hradílek** jhradilek@redhat.com

Updated Network Interfaces.

Revision 1-0 **Tue May 29 2012** **Jaromír Hradílek** jhradilek@redhat.com

Fedora 17 release of the *System Administrator's Guide*.

Index

Symbols

.fetchmailrc, 163
 server options, 164
 user options, 165
.htaccess, 113, 117
 (see also Apache HTTP Server)
.htpasswd, 113
 (see also Apache HTTP Server)
.procmailrc, 167
/dev/oprofile/, 403
/dev/shm, 313
/run, 313
/sys/fs/cgroup, 313
/var/spool/anacron , 378
/var/spool/cron , 380

(see OProfile)

A

adding
 group, 33
 user, 30
anacron, 377
 anacron configuration file, 378
 user-defined tasks, 378
anacrontab , 378
Apache HTTP Server
 additional resources
 installable documentation, 149
 installed documentation, 149
 useful websites, 150
 checking configuration, 110
 checking status, 109
 directives
 <Directory>, 110
 <IfDefine>, 111
 <IfModule>, 111
 <Location>, 111
 <Proxy>, 112
 <VirtualHost>, 112
 AccessFileName, 113
 Action, 113
 AddDescription, 113
 AddEncoding, 114
 AddHandler, 114
 AddIcon, 114
 AddIconByEncoding, 115
 AddIconByType, 115
 AddLanguage, 115
 AddType, 116
 Alias, 116

Allow, 116
AllowOverride, 117
BrowserMatch, 117
CacheDefaultExpire, 118
CacheDisable, 118
CacheEnable, 118
CacheLastModifiedFactor, 119
CacheMaxExpire, 119
CacheNegotiatedDocs, 119
CacheRoot, 119
CustomLog, 120
DefaultIcon, 120
DefaultType, 120
Deny, 121
DirectoryIndex, 121
DocumentRoot, 121
ErrorDocument, 121
ErrorLog, 122
ExtendedStatus, 122
Group, 122
HeaderName, 123
HostnameLookups, 123
Include, 124
IndexIgnore, 124
IndexOptions, 124
KeepAlive, 125
KeepAliveTimeout, 126
LanguagePriority, 126
Listen, 126
LoadModule, 127
LogFormat, 127
LogLevel, 128
MaxClients, 138, 138
MaxKeepAliveRequests, 128
MaxSpareServers, 138
MaxSpareThreads, 139
MinSpareServers, 139
MinSpareThreads, 139
NameVirtualHost, 129
Options, 129
Order, 130
PidFile, 130
ProxyRequests, 130
ReadmeName, 131
Redirect, 131
ScriptAlias, 132
ServerAdmin, 132
ServerName, 132
ServerRoot, 133
ServerSignature, 133
ServerTokens, 134
SetEnvIf, 137
StartServers, 139
SuexecUserGroup, 134

- ThreadsPerChild, 140
- Timeout, 134
- TypesConfig, 135
- UseCanonicalName, 135
- User, 136
- UserDir, 136
- directories
 - /etc/httpd/, 133
 - /etc/httpd/conf.d/, 110, 124
 - /usr/lib/httpd/modules/, 127, 140
 - /usr/lib64/httpd/modules/, 127, 140
 - /var/cache/mod_proxy/, 120
 - /var/www/cgi-bin/, 132
 - /var/www/html/, 121
 - /var/www/icons/, 116
 - ~/public_html/, 136
- files
 - .htaccess, 113, 117
 - .htpasswd, 113
 - /etc/httpd/conf.d/ssl.conf, 137, 143
 - /etc/httpd/conf/httpd.conf, 110, 110, 138
 - /etc/httpd/logs/access_log, 120
 - /etc/httpd/logs/error_log, 122
 - /etc/httpd/run/httpd.pid, 130
 - /etc/mime.types, 135
- modules
 - developing, 140
 - loading, 140
 - mod_rewrite, 131
 - mod_ssl, 141
 - mod_userdir, 108
- restarting, 109
- SSL server
 - certificate, 142, 144, 145
 - certificate authority, 142
 - private key, 142, 144, 145
 - public key, 142
- starting, 108
- stopping, 109
- version 2.4
 - changes, 105
 - updating from version 2.2, 107
- virtual host, 141
- at , 382
 - additional resources, 386
- Automated Tasks, 377

B

- batch , 382
 - additional resources, 386
- blkid, 310
- boot loader
 - GRUB 2 boot loader, 411
 - verifying, 436

- boot media, 432

C

- ch-email .fetchmailrc
 - global options, 164
- chage command
 - forcing password expiration with, 33
- Configuration File Changes, 47
- CPU usage, 308
- cron, 377
 - additional resources, 386
 - cron configuration file, 380
 - user-defined tasks, 380
- crontab , 380
- CUPS (see Printer Configuration)

D

- df, 312
- directory server (see OpenLDAP)
- DNF
 - Additional Resources, 61
 - configuring DNF and DNF repositories, 57
 - displaying packages
 - dnf info, 52
 - displaying packages with DNF
 - dnf info, 51
 - DNF repositories
 - configuring DNF and DNF repositories, 57
 - installing a package group with DNF, 53
 - installing with DNF, 52
 - listing packages from a single repository with DNF
 - dnf repository-packages, 51
 - listing packages with DNF
 - dnf group list, 50
 - dnf list, 48
 - dnf list all, 49
 - dnf list available, 50
 - dnf list installed, 50
 - dnf repolist, 51
 - Glob expressions, 49
 - packages and package groups, 48
 - repository, 60
 - searching for packages with DNF
 - dnf search, 48
 - searching packages with DNF
 - dnf search, 48
 - setting [main] options, 57
 - setting [repository] options, 58
 - uninstalling packages with DNF, 54
 - dnf remove package_name, 54
 - variables, 59
- DNF Updates

- checking for updates, 45
- updating a single package, 46
- updating all packages and dependencies, 47
- updating packages, 46

documentation

- finding installed, 464

drivers (see kernel module)

du, 313

E

ECDSA keys

- generating, 79

email

- additional resources, 175
 - installed documentation, 175
 - related books, 176
 - useful websites, 176

Fetchmail, 162

mail server

- Dovecot, 153

Postfix, 155

Procmail, 167

program classifications, 154

protocols, 151

- IMAP, 152
- POP, 151
- SMTP, 151

security, 173

- clients, 174
- servers, 174

Sendmail, 157

spam

- filtering out, 172

types

- Mail Delivery Agent, 154
- Mail Transport Agent, 154
- Mail User Agent, 155

expiration of password, forcing, 33

F

Fedora installation media

- installable packages, 462

feedback

- contact information for this manual, xx

Fetchmail, 162

- additional resources, 175
- command options, 165
 - informational, 166
 - special, 166
- configuration options, 163
 - global options, 164
 - server options, 164
 - user options, 165

file systems, 309

findmnt, 311

free, 306

FTP, 221

- (see also vsftpd)
- active mode, 221
- command port, 221
- data port, 221
- definition of, 221
- introducing, 221
- passive mode, 221
- server software
 - Red Hat Content Accelerator , 222
 - vsftpd , 222

G

gnome-system-log (see System Log)

gnome-system-monitor, 305, 307, 308, 314

GnuPG

- checking RPM package signatures, 463

group configuration

- groupadd, 33
- viewing list of groups, 28

groups (see group configuration)

- additional resources, 36
- installed documentation, 37

GID, 27

introducing, 27

shared directories, 36

tools for management of

- groupadd, 27, 29
- user private, 27

GRUB 2

- configuring GRUB 2, 411
- customizing GRUB 2, 411
- reinstalling GRUB 2, 411

GRUB 2 boot loader

- configuration file, 437
- configuring, 437

GUI, 3

H

hardware

- viewing, 315

HTTP server (see Apache HTTP Server)

httpd (see Apache HTTP Server)

I

information

- about your system, 303

initial RAM disk image

- verifying, 434
- IBM eServer System i, 436

- initial RPM repositories
 - installable packages, 462
- insmod, 445
 - (see also kernel module)
- installing the kernel, 431

K

- kernel
 - downloading, 433
 - installing kernel packages, 431, 431
 - kernel packages, 431
 - package, 431
 - performing kernel upgrade, 433
 - RPM package, 431
 - upgrade kernel available, 433
 - Security Advisories, 433
 - via Fedora Update System, 433
 - upgrading
 - preparing, 432
 - working boot media, 432
 - upgrading the kernel, 431
- kernel module
 - definition, 441
 - directories
 - /etc/modules-load.d/, 447
 - /lib/modules/kernel_version/kernel/drivers/ , 444
 - files
 - /proc/modules, 442
 - listing
 - currently loaded modules, 441
 - module information, 442
 - loading
 - at the boot time, 447
 - for the current session, 444
 - module parameters
 - supplying, 446
 - unloading, 445
 - utilities
 - insmod, 445
 - lsmod, 441
 - modinfo, 442
 - modprobe, 444, 445
 - rmmod, 446
- kernel package
 - kernel
 - for single, multicore and multiprocessor systems, 431
 - kernel-devel
 - kernel headers and makefiles, 431
 - kernel-headers
 - C header files files, 431
 - linux-firmware
 - firmware files, 431

- perf
 - firmware files, 431
- kernel upgrading
 - preparing, 432
- keyboard configuration, 15
 - layout, 17

L

- LDAP (see OpenLDAP)
- localectl (see keyboard configuration)
- log files, 331
 - (see also System Log)
 - description, 331
 - locating, 331
 - monitoring, 373
 - rotating, 331
 - rsyslogd daemon, 331
 - viewing, 369
- logrotate, 331
- lsblk, 309
- lscpu, 317
- lsmod, 441
 - (see also kernel module)
- lspci, 315
- ls pcmcia, 316
- lsusb, 315

M

- Mail Delivery Agent (see email)
- Mail Transport Agent (see email) (see MTA)
- Mail Transport Agent Switcher, 166
- Mail User Agent, 166 (see email)
- MDA (see Mail Delivery Agent)
- memory usage, 306
- modinfo, 442
 - (see also kernel module)
- modprobe, 444, 445
 - (see also kernel module)
- module (see kernel module)
- module parameters (see kernel module)
- MTA (see Mail Transport Agent)
 - setting default, 166
 - switching with Mail Transport Agent Switcher, 166
- MUA, 166 (see Mail User Agent)

N

- net program, 216
- nmblookup program, 217

O

- opannotate (see OProfile)
- opcontrol (see OProfile)

OpenLDAP

- backends, 192
 - checking status, 194
 - client applications, 183
 - configuration
 - database, 187
 - global, 184
 - overview, 181
 - TLS, 188, 192, 192
 - directives
 - olcAllows, 184
 - olcConnMaxPending, 185
 - olcConnMaxPendingAuth, 185
 - olcDisallows, 185
 - olcIdleTimeout, 186
 - olcLogFile, 186
 - olcReadOnly, 187
 - olcReferral, 186
 - olcRootDN, 187
 - olcRootPW, 187
 - olcSuffix, 188
 - olcWriteTimeout, 186
 - directories
 - /etc/openldap/slapd.d/, 184, 192, 192
 - /etc/openldap/slapd.d/cn=config/
 - cn=schema/, 188
 - features, 180
 - files
 - /etc/openldap/ldap.conf, 184, 188
 - /etc/openldap/slapd.d/cn=config.ldif, 184, 188
 - /etc/openldap/slapd.d/cn=config/
 - olcDatabase={1}bdb.ldif, 187
 - installation, 181
 - migrating authentication information, 194
 - modules, 192
 - packages, 181
 - replication, 192
 - restarting, 194
 - running, 193
 - schema, 188
 - security, 188
 - stopping, 193
 - terminology
 - attribute, 180
 - entry, 180
 - LDIF, 180
 - utilities, 182, 183
- ## OpenSSH, 71, 72
- (see also SSH)
- additional resources, 95
 - client, 91
 - scp, 92
 - sftp, 93

- ssh, 91
 - ECDSA keys
 - generating, 79
 - RSA keys
 - generating, 78
 - server, 76
 - starting, 76
 - stopping, 76
 - ssh-agent, 80
 - ssh-keygen
 - ECDSA, 79
 - RSA, 78
 - using key-based authentication, 77
- ## OpenSSL
- additional resources, 95
 - SSL (see SSL)
 - TLS (see TLS)
- ## ophelp, 394
- ## opreport (see OProfile)
- ## OProfile, 387
- /dev/oprofile/, 403
 - additional resources, 407
 - configuring, 391
 - separating profiles, 396
 - events
 - sampling rate, 395
 - setting, 392
 - Java, 403
 - monitoring the kernel, 391
 - opannotate, 402
 - opcontrol, 391
 - no-vmlinux, 392
 - start, 397
 - vmlinux=, 392
 - ophelp, 394
 - opreport, 399, 401
 - on a single executable, 400
 - oprofiled, 397
 - log file, 397
 - overview of tools, 387
 - reading data, 398
 - saving data, 397
 - starting, 397
 - SystemTap, 406
 - unit mask, 395
- ## oprofiled (see OProfile)
- ## oprof_start, 404
- ## OS/400 boot loader
- configuration file, 438
 - configuring, 438

P

- package
 - kernel RPM, 431

packages

- dependencies, 459
- determining file ownership with, 464
- displaying packages
 - dnf info, 52
- displaying packages with DNF
 - dnf info, 51
- DNF instead of RPM, 455
- Fedora installation media, 462
- finding deleted files from, 464
- finding Fedora RPM packages, 462
- initial RPM repositories, 462
- installing a package group with DNF, 53
- installing RPM, 456
- installing with DNF, 52
- kernel
 - for single, multicore and multiprocessor systems, 431
- kernel-devel
 - kernel headers and makefiles, 431
- kernel-headers
 - C header files files, 431
- linux-firmware
 - firmware files, 431
- listing packages from a single repository with DNF
 - dnf repository-packages, 51
- listing packages with DNF
 - dnf group list, 50
 - dnf list all, 49
 - dnf list available, 50
 - dnf list installed, 50
 - dnf repolist, 51
 - dnf search, 48
 - Glob expressions, 49
- locating documentation for, 464
- obtaining list of files, 464
- packages and package groups, 48
- perf
 - firmware files, 431
- querying uninstalled, 464
- removing, 459
- RPM, 455
 - already installed, 458
 - configuration file changes, 459
 - conflict, 458
 - failed dependencies, 459
 - freshening, 460
 - pristine sources, 456
 - querying, 461
 - removing, 459
 - source and binary packages, 455
 - tips, 464
 - uninstalling, 459

- verifying, 461
- searching for packages with DNF
 - dnf search, 48
- searching packages with DNF
 - dnf search, 48
- uninstalling packages with DNF, 54
 - dnf remove package_name, 54
- upgrading RPM, 456
- partx, 311
- password
 - aging, 33
 - expire, 33
- passwords
 - shadow, 27
- pdbedit program, 217
- Postfix, 155
 - default installation, 156
- postfix, 166
- Printer Configuration
 - CUPS, 234
 - IPP Printers, 237
 - LDP/LPR Printers, 238
 - Local Printers, 235
 - New Printer, 235
 - Print Jobs, 250
 - Samba Printers, 239
 - Settings, 245
 - Sharing Printers, 246
- printers (see Printer Configuration)
- processes, 303
- Procmail, 167
 - additional resources, 175
 - configuration, 167
 - recipes, 168
 - delivering, 169
 - examples, 171
 - flags, 169
 - local lockfiles, 170
 - non-delivering, 169
 - SpamAssassin, 172
 - special actions, 170
 - special conditions, 170
- ps, 303

R

- RAM, 306
- rcp, 92
- rmmod, 446
 - (see also kernel module)
- rpcclient program, 218
- RPM, 455
 - additional resources, 464
 - already installed, 458
 - basic modes, 456

- checking package signatures, 463
- configuration file changes, 459
 - conf.rpmsave, 459
- conflicts, 458
- dependencies, 459
- design goals, 455
 - powerful querying, 456
 - system verification, 456
 - upgradability, 455
- determining file ownership with, 464
- documentation with, 464
- failed dependencies, 459
- file conflicts
 - resolving, 458
- file name, 456
- finding and verifying RPM packages, 462
- finding deleted files with, 464
- finding Fedora RPM packages, 462
- freshening, 460
- GnuPG, 463
- installing, 456
- online documentation, 465
- querying, 461
- querying for file list, 464
- querying uninstalled packages, 464
- see also, 465
- tips, 464
- uninstalling, 459
- upgrading, 456
- verification, 462, 462
- verifying, 461
- website, 465

RPM Package Manager (see RPM)

RSA keys

- generating, 78

rsyslog, 331

- actions, 335
- configuration, 331
- debugging, 363
- filters, 332
- global directives, 343
- log rotation, 344
- modules, 356
- new configuration format, 345
- queues, 347
- rulesets, 346
- templates, 340

S

Samba (see Samba)

- Abilities, 197
- Account Information Databases, 213
 - ldapsam, 213
 - ldapsam_compat, 213

- mysqlsam, 213
- Plain Text, 213
- smbpasswd, 213
- tdbsam, 213
- xmlsam, 213

Additional Resources, 220

- installed documentation, 220
- useful websites, 221

Backward Compatible Database Back Ends, 213

Browsing, 214

configuration, 201, 201

- default, 201

CUPS Printing Support, 215

- CUPS smb.conf, 215

daemon

- nmbd, 198
- overview, 198
- smbd, 198
- winbindd, 198

encrypted passwords, 202

graphical configuration, 201

Introduction, 197

Network Browsing, 214

- Domain Browsing, 214
- WINS, 215

New Database Back Ends, 213

Programs, 216

- net, 216
- nmblookup, 217
- pdbedit, 217
- rpcclient, 218
- smbcacls, 218
- smbclient, 218
- smbcontrol, 218
- smbpasswd, 219
- smbpool, 219
- smbstatus, 219
- smbtar, 219
- testparm, 219
- wbinfo, 220

Reference, 197

Samba Printers, 239

Security Modes, 211, 211

- Active Directory Security Mode, 212
- Domain Security Mode, 212
- Share-Level Security, 213
- User Level Security, 211

Server Types, 203

server types

- Domain Controller, 208
- Domain Member, 206
- Stand Alone, 203

service

- conditional restarting, 202
 - reloading, 202
 - restarting, 202
 - starting, 202
 - stopping, 202
 - share
 - connecting to via the command line, 200
 - connecting to with Nautilus, 199
 - mounting, 200
 - smb.conf, 203
 - Active Directory Member Server example, 206
 - Anonymous Print Server example, 205
 - Anonymous Read Only example, 204
 - Anonymous Read/Write example, 204
 - NT4-style Domain Member example, 207
 - PDC using Active Directory, 211
 - PDC using tdbsam, 209
 - Secure File and Print Server example, 205
 - smbclient, 200
 - WINS, 215
 - with Windows NT 4.0, 2000, ME, and XP, 202
 - scp (see OpenSSH)
 - Sendmail, 157
 - additional resources, 175
 - aliases, 160
 - common configuration changes, 160
 - default installation, 158
 - LDAP and, 162
 - limitations, 158
 - masquerading, 160
 - purpose, 158
 - spam, 161
 - with UUCP, 160
 - sendmail, 166
 - services configuration, 65
 - sssystemctl , 66
 - systemctl , 65
 - sftp (see OpenSSH)
 - shadow passwords
 - overview of, 27
 - slapd (see OpenLDAP)
 - smbcacls program, 218
 - smbclient, 200
 - smbclient program, 218
 - smbcontrol program, 218
 - smbpasswd program, 219
 - smbspool program, 219
 - smbstatus program, 219
 - smbtar program, 219
 - SpamAssassin
 - using with Procmail, 172
 - ssh (see OpenSSH)
 - SSH protocol
 - authentication, 74
 - configuration files, 75
 - system-wide configuration files, 75
 - user-specific configuration files, 75
 - connection sequence, 73
 - features, 72
 - insecure protocols, 77
 - layers
 - channels, 74
 - transport layer, 73
 - port forwarding, 94
 - requiring for remote login, 77
 - security risks, 71
 - version 1, 72
 - version 2, 72
 - X11 forwarding, 94
 - ssh-agent, 80
 - SSL, 141
 - (see also Apache HTTP Server)
 - SSL server (see Apache HTTP Server)
 - stunnel, 174
 - system analysis
 - OProfile (see OProfile)
 - system information
 - cpu usage, 308
 - file systems, 309
 - /dev/shm, 313
 - /run, 313
 - /sys/fs/cgroup, 313
 - gathering, 303
 - hardware, 315
 - memory usage, 306
 - processes, 303
 - currently running, 304
 - System Log
 - filtering, 370
 - monitoring, 373
 - refresh rate, 370
 - searching, 370
 - System Monitor, 305, 307, 308, 314
 - systemctl (see services configuration)
- T**
- testparm program, 219
 - the Users settings tool (see user configuration)
 - TLS, 141
 - (see also Apache HTTP Server)
 - top, 304
- U**
- user configuration
 - command line configuration
 - chage, 33

- passwd, 30
- useradd, 30
- password
 - forcing expiration of, 33
 - viewing list of users, 28
- user private groups (see groups)
 - and shared directories, 36
- useradd command
 - user account creation using, 30
- users (see user configuration)
 - additional resources, 36
 - installed documentation, 37
 - introducing, 27
 - tools for management of
 - the Users setting tool, 29
 - useradd, 29
- UID, 27

V

- virtual host (see Apache HTTP Server)
- vsftpd , 222
 - (see also FTP)
 - additional resources, 233
 - installed documentation, 233
 - useful websites, 234
 - condrestart, 223
 - configuration file
 - /etc/vsftpd/vsftpd.conf , 224
 - access controls, 226
 - anonymous user options, 227
 - daemon options, 225
 - directory options, 229
 - file transfer options, 229
 - format of, 224
 - local user options, 228
 - logging options, 230
 - login options, 226
 - network options, 231
- multihome configuration, 224
- restarting, 223
- RPM
 - files installed by, 223
- security features, 222
- starting, 223
- starting multiple copies of, 224
- status, 223
- stopping, 223

W

- wbinfo program, 220
- web server (see Apache HTTP Server)
- Windows 2000
 - connecting to shares using Samba, 202

- Windows 98
 - connecting to shares using Samba, 202
- Windows ME
 - connecting to shares using Samba, 202
- Windows NT 4.0
 - connecting to shares using Samba, 202
- Windows XP
 - connecting to shares using Samba, 202

X

- X.500 (see OpenLDAP)
- X.500 Lite (see OpenLDAP)
