



Red Hat Enterprise Linux 6

Deployment Guide

Deployment, Configuration and Administration of Red Hat Enterprise Linux 6

Red Hat Enterprise Linux 6 Deployment Guide

Deployment, Configuration and Administration of Red Hat Enterprise Linux 6

Marie Doleželová
Red Hat Customer Content Services
mdolezel@redhat.com

Mirek Jahoda
Red Hat Customer Content Services
mjahoda@redhat.com

Maxim Svistunov
Red Hat Customer Content Services

Stephen Wadeley
Red Hat Customer Content Services

Tomáš Čapek
Red Hat Customer Content Services

Robert Krátký
Red Hat Customer Content Services

Jana Heves
Red Hat Customer Content Services

Jaromír Hradílek
Red Hat Customer Content Services

Douglas Silas
Red Hat Customer Content Services

Barbora Ančincová
Red Hat Customer Content Services

Petr Kovář
Red Hat Customer Content Services

Jiří Herrmann
Red Hat Customer Content Services

Peter Ondrejka
Red Hat Customer Content Services

Petr Bokoč
Red Hat Customer Content Services

Martin Prpič
Red Hat Product Security

Legal Notice

Red Hat Customer Content Services
Copyright © 2010–2017 Red Hat, Inc.

Eva Kopalová

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

Red Hat waives the right to enforce, and agrees not to assert, Red Hat Customer Content Services' trademark rights to the extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Red Hat Glassfish, Red Hat JBoss Seam, Red Hat JBoss Seam are trademarks of Red Hat, Inc., registered in the United States and other countries.

Florian Nadge

Red Hat 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.

Red Hat Customer Content Services

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

Red Hat Customer Content Services

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

Red Hat Customer Content Services

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

Red Hat Customer Content Services

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

All other trademarks are the property of their respective owners.

Abstract

The Deployment Guide documents relevant information regarding the deployment, configuration and administration of Red Hat Enterprise Linux 6. It is oriented towards system administrators with a basic understanding of the system.

Table of Contents

PART I. BASIC SYSTEM CONFIGURATION	17
CHAPTER 1. KEYBOARD CONFIGURATION	18
1.1. CHANGING THE KEYBOARD LAYOUT	18
1.2. ADDING THE KEYBOARD LAYOUT INDICATOR	20
1.3. SETTING UP A TYPING BREAK	22
CHAPTER 2. DATE AND TIME CONFIGURATION	24
2.1. DATE/TIME PROPERTIES TOOL	24
2.1.1. Date and Time Properties	24
2.1.2. Network Time Protocol Properties	25
2.1.3. Time Zone Properties	26
2.2. COMMAND LINE CONFIGURATION	27
2.2.1. Date and Time Setup	28
2.2.2. Network Time Protocol Setup	28
CHAPTER 3. MANAGING USERS AND GROUPS	31
3.1. WHAT USERS AND GROUPS ARE	31
3.2. MANAGING USERS VIA THE USER MANAGER APPLICATION	32
3.2.1. Viewing Users	32
3.2.2. Adding a New User	33
3.2.3. Modifying User Properties	34
3.3. MANAGING GROUPS VIA THE USER MANAGER APPLICATION	35
3.3.1. Viewing Groups	35
3.3.2. Adding a New Group	36
3.3.3. Modifying Group Properties	37
3.4. MANAGING USERS VIA COMMAND-LINE TOOLS	37
3.4.1. Creating Users	37
3.4.2. Attaching New Users to Groups	40
3.4.3. Updating Users' Authentication	40
3.4.4. Modifying User Settings	42
3.4.5. Deleting Users	43
3.4.6. Displaying Comprehensive User Information	43
3.5. MANAGING GROUPS VIA COMMAND-LINE TOOLS	46
3.5.1. Creating Groups	46
3.5.2. Attaching Users to Groups	47
3.5.3. Updating Group Authentication	47
3.5.4. Modifying Group Settings	48
3.5.5. Deleting Groups	48
3.6. ADDITIONAL RESOURCES	48
3.6.1. Installed Documentation	49
CHAPTER 4. GAINING PRIVILEGES	50
4.1. THE SU COMMAND	50
4.2. THE SUDO COMMAND	51
4.3. ADDITIONAL RESOURCES	52
Installed Documentation	52
Online Documentation	52
CHAPTER 5. CONSOLE ACCESS	54
5.1. DISABLING CONSOLE PROGRAM ACCESS FOR NON-ROOT USERS	54
5.2. DISABLING REBOOTING USING CTRL+ALT+DEL	55

PART II. SUBSCRIPTION AND SUPPORT	56
CHAPTER 6. REGISTERING THE SYSTEM AND MANAGING SUBSCRIPTIONS	57
6.1. REGISTERING THE SYSTEM AND ATTACHING SUBSCRIPTIONS	57
6.2. MANAGING SOFTWARE REPOSITORIES	58
6.3. REMOVING SUBSCRIPTIONS	58
6.4. ADDITIONAL RESOURCES	59
Installed Documentation	59
Related Books	59
Online Resources	59
See Also	60
CHAPTER 7. ACCESSING SUPPORT USING THE RED HAT SUPPORT TOOL	61
7.1. INSTALLING THE RED HAT SUPPORT TOOL	61
7.2. REGISTERING THE RED HAT SUPPORT TOOL USING THE COMMAND LINE	61
7.3. USING THE RED HAT SUPPORT TOOL IN INTERACTIVE SHELL MODE	61
7.4. CONFIGURING THE RED HAT SUPPORT TOOL	61
7.4.1. Saving Settings to the Configuration Files	62
7.5. OPENING AND UPDATING SUPPORT CASES USING INTERACTIVE MODE	63
7.6. VIEWING SUPPORT CASES ON THE COMMAND LINE	65
7.7. ADDITIONAL RESOURCES	65
PART III. INSTALLING AND MANAGING SOFTWARE	66
CHAPTER 8. YUM	67
8.1. CHECKING FOR AND UPDATING PACKAGES	67
8.1.1. Checking For Updates	67
8.1.2. Updating Packages	68
8.1.3. Preserving Configuration File Changes	70
8.1.4. Upgrading the System Off-line with ISO and Yum	70
8.2. PACKAGES AND PACKAGE GROUPS	72
8.2.1. Searching Packages	72
8.2.2. Listing Packages	73
8.2.3. Displaying Package Information	75
8.2.4. Installing Packages	76
8.2.5. Removing Packages	78
8.3. WORKING WITH TRANSACTION HISTORY	79
8.3.1. Listing Transactions	79
8.3.2. Examining Transactions	82
8.3.3. Reverting and Repeating Transactions	83
8.3.4. Completing Transactions	84
8.3.5. Starting New Transaction History	84
8.4. CONFIGURING YUM AND YUM REPOSITORIES	84
8.4.1. Setting [main] Options	85
8.4.2. Setting [repository] Options	88
8.4.3. Using Yum Variables	90
8.4.4. Viewing the Current Configuration	91
8.4.5. Adding, Enabling, and Disabling a Yum Repository	92
8.4.6. Creating a Yum Repository	93
8.4.7. Working with Yum Cache	94
8.4.8. Adding the Optional and Supplementary Repositories	95
8.5. YUM PLUG-INS	96
8.5.1. Enabling, Configuring, and Disabling Yum Plug-ins	96
8.5.2. Installing Additional Yum Plug-ins	97

8.5.3. Plug-in Descriptions	97
8.6. ADDITIONAL RESOURCES	101
Installed Documentation	101
Online Resources	102
See Also	102
CHAPTER 9. PACKAGEKIT	103
9.1. UPDATING PACKAGES WITH SOFTWARE UPDATE	103
Setting the Update-Checking Interval	104
9.2. USING ADD/REMOVE SOFTWARE	105
9.2.1. Refreshing Software Sources (Yum Repositories)	105
9.2.2. Finding Packages with Filters	106
9.2.3. Installing and Removing Packages (and Dependencies)	108
9.2.4. Installing and Removing Package Groups	109
9.2.5. Viewing the Transaction Log	110
9.3. PACKAGEKIT ARCHITECTURE	111
9.4. ADDITIONAL RESOURCES	112
Installed Documentation	112
Online Documentation	113
See Also	113
PART IV. NETWORKING	114
CHAPTER 10. NETWORKMANAGER	115
10.1. THE NETWORKMANAGER DAEMON	115
10.2. INTERACTING WITH NETWORKMANAGER	115
10.2.1. Connecting to a Network	116
10.2.2. Configuring New and Editing Existing Connections	117
10.2.3. Connecting to a Network Automatically	118
10.2.4. User and System Connections	119
10.3. ESTABLISHING CONNECTIONS	120
10.3.1. Establishing a Wired (Ethernet) Connection	120
10.3.2. Establishing a Wireless Connection	123
10.3.3. Establishing a Mobile Broadband Connection	128
10.3.4. Establishing a VPN Connection	131
10.3.5. Establishing a DSL Connection	134
10.3.6. Establishing a Bond Connection	136
10.3.7. Establishing a VLAN Connection	140
10.3.8. Establishing an IP-over-InfiniBand (IPoIB) Connection	142
10.3.9. Configuring Connection Settings	144
CHAPTER 11. NETWORK INTERFACES	153
11.1. NETWORK CONFIGURATION FILES	153
11.1.1. Setting the Host Name	154
11.2. INTERFACE CONFIGURATION FILES	154
11.2.1. Ethernet Interfaces	154
11.2.2. Specific ifcfg Options for Linux on System z	160
11.2.3. Required ifcfg Options for Linux on System z	161
11.2.4. Channel Bonding Interfaces	161
11.2.5. Configuring a VLAN over a Bond	164
11.2.6. Network Bridge	171
11.2.7. Setting Up 802.1Q VLAN Tagging	175
11.2.8. Alias and Clone Files	176
11.2.9. Dialup Interfaces	177

11.2.10. Other Interfaces	179
11.3. INTERFACE CONTROL SCRIPTS	180
11.4. STATIC ROUTES AND THE DEFAULT GATEWAY	181
Configuring Static Routes Using the Command Line	182
Configuring The Default Gateway	182
11.5. CONFIGURING STATIC ROUTES IN IFCFG FILES	183
11.5.1. Static Routes Using the IP Command Arguments Format	183
11.5.2. Network/Netmask Directives Format	184
11.6. CONFIGURING IPV6 TOKENIZED INTERFACE IDENTIFIERS	184
11.7. NETWORK FUNCTION FILES	185
11.8. ETHTOOL	186
11.9. CONFIGURING NETCONSOLE	193
Configuring a Listening Machine	193
Configuring a Sending Machine	194
11.10. ADDITIONAL RESOURCES	194
Installed Documentation	194
Online Resources	194
See Also	195
PART V. INFRASTRUCTURE SERVICES	196
CHAPTER 12. SERVICES AND DAEMONS	197
12.1. CONFIGURING THE DEFAULT RUNLEVEL	197
12.2. CONFIGURING THE SERVICES	198
12.2.1. Using the Service Configuration Utility	198
12.2.2. Using the ntsysv Utility	200
12.2.3. Using the chkconfig Utility	202
12.3. RUNNING SERVICES	204
12.3.1. Determining the Service Status	204
12.3.2. Starting a Service	204
12.3.3. Stopping a Service	205
12.3.4. Restarting a Service	205
12.4. ADDITIONAL RESOURCES	205
12.4.1. Installed Documentation	205
12.4.2. Related Books	205
CHAPTER 13. CONFIGURING AUTHENTICATION	206
13.1. CONFIGURING SYSTEM AUTHENTICATION	206
13.1.1. Launching the Authentication Configuration Tool UI	206
13.1.2. Selecting the Identity Store for Authentication	207
13.1.3. Configuring Alternative Authentication Features	214
13.1.4. Configuring Authentication from the Command Line	216
13.1.5. Using Custom Home Directories	219
13.2. USING AND CACHING CREDENTIALS WITH SSSD	220
13.2.1. About SSSD	220
13.2.2. Setting up the sssd.conf File	221
13.2.3. Starting and Stopping SSSD	223
13.2.4. SSSD and System Services	223
13.2.5. Configuring Services: NSS	224
13.2.6. Configuring Services: PAM	227
13.2.7. Configuring Services: autofs	229
13.2.8. Configuring Services: sudo	232
13.2.9. Configuring Services: OpenSSH and Cached Keys	235
13.2.10. SSSD and Identity Providers (Domains)	237

13.2.11. Creating Domains: LDAP	240
13.2.12. Creating Domains: Identity Management (IdM)	243
13.2.13. Creating Domains: Active Directory	246
13.2.14. Configuring Domains: Active Directory as an LDAP Provider (Alternative)	252
13.2.15. Domain Options: Setting Username Formats	256
13.2.16. Domain Options: Enabling Offline Authentication	257
13.2.17. Domain Options: Setting Password Expirations	259
13.2.18. Domain Options: Using DNS Service Discovery	260
13.2.19. Domain Options: Using IP Addresses in Certificate Subject Names (LDAP Only)	262
13.2.20. Creating Domains: Proxy	263
13.2.21. Creating Domains: Kerberos Authentication	265
13.2.22. Creating Domains: Access Control	268
13.2.23. Creating Domains: Primary Server and Backup Servers	270
13.2.24. Installing SSSD Utilities	270
13.2.25. SSSD and UID and GID Numbers	271
13.2.26. Creating Local System Users	271
13.2.27. Seeding Users into the SSSD Cache During Kickstart	272
13.2.28. Managing the SSSD Cache	273
13.2.29. Downgrading SSSD	274
13.2.30. Using NSCD with SSSD	275
13.2.31. Troubleshooting SSSD	275
CHAPTER 14. OPENSSH	283
14.1. THE SSH PROTOCOL	283
14.1.1. Why Use SSH?	283
14.1.2. Main Features	283
14.1.3. Protocol Versions	284
14.1.4. Event Sequence of an SSH Connection	284
14.2. CONFIGURING OPENSSH	286
14.2.1. Configuration Files	286
14.2.2. Starting an OpenSSH Server	288
14.2.3. Requiring SSH for Remote Connections	289
14.2.4. Using Key-Based Authentication	289
14.3. USING OPENSSH CERTIFICATE AUTHENTICATION	295
14.3.1. Introduction to SSH Certificates	295
14.3.2. Support for SSH Certificates	295
14.3.3. Creating SSH CA Certificate Signing Keys	295
14.3.4. Distributing and Trusting SSH CA Public Keys	298
14.3.5. Creating SSH Certificates	299
14.3.6. Signing an SSH Certificate Using a PKCS#11 Token	303
14.3.7. Viewing an SSH CA Certificate	304
14.3.8. Revoking an SSH CA Certificate	305
14.4. OPENSSH CLIENTS	305
14.4.1. Using the ssh Utility	305
14.4.2. Using the scp Utility	306
14.4.3. Using the sftp Utility	307
14.5. MORE THAN A SECURE SHELL	308
14.5.1. X11 Forwarding	308
14.5.2. Port Forwarding	308
14.6. ADDITIONAL RESOURCES	309
14.6.1. Installed Documentation	309
14.6.2. Useful Websites	310

CHAPTER 15. TIGERVNC	311
15.1. VNC SERVER	311
15.1.1. Installing VNC Server	311
15.1.2. Configuring VNC Server	311
15.1.3. Starting VNC Server	313
15.1.4. Terminating a VNC Session	314
15.2. SHARING AN EXISTING DESKTOP	314
15.3. USING A VNC VIEWER	315
15.3.1. Installing the VNC Viewer	315
15.3.2. Connecting to a VNC Server	315
15.3.3. Connecting to VNC Server Using SSH	318
15.4. ADDITIONAL RESOURCES	318
Installed Documentation	318
PART VI. SERVERS	319
CHAPTER 16. DHCP SERVERS	320
16.1. WHY USE DHCP?	320
16.2. CONFIGURING A DHCPV4 SERVER	320
16.2.1. Configuration File	320
16.2.2. Lease Database	323
16.2.3. Starting and Stopping the Server	324
16.2.4. DHCP Relay Agent	324
16.3. CONFIGURING A DHCPV4 CLIENT	325
16.4. CONFIGURING A MULTIHOME DHCP SERVER	325
16.4.1. Host Configuration	327
16.5. DHCP FOR IPV6 (DHCPV6)	328
16.5.1. Configuring a DHCPv6 Server	329
16.5.2. Configuring a DHCPv6 Client	329
16.6. ADDITIONAL RESOURCES	330
16.6.1. Installed Documentation	330
CHAPTER 17. DNS SERVERS	331
17.1. INTRODUCTION TO DNS	331
17.1.1. Nameserver Zones	331
17.1.2. Nameserver Types	331
17.1.3. BIND as a Nameserver	332
17.2. BIND	332
17.2.1. Configuring the named Service	332
17.2.2. Editing Zone Files	341
17.2.3. Using the rndc Utility	348
17.2.4. Using the dig Utility	351
17.2.5. Advanced Features of BIND	353
17.2.6. Common Mistakes to Avoid	354
17.2.7. Additional Resources	355
CHAPTER 18. WEB SERVERS	357
18.1. THE APACHE HTTP SERVER	357
18.1.1. New Features	357
18.1.2. Notable Changes	357
18.1.3. Updating the Configuration	357
18.1.4. Running the httpd Service	358
18.1.5. Editing the Configuration Files	359
18.1.6. Working with Modules	393

18.1.7. Setting Up Virtual Hosts	394
18.1.8. Setting Up an SSL Server	394
18.1.9. Enabling the mod_ssl Module	395
18.1.10. Enabling the mod_nss Module	398
18.1.11. Using an Existing Key and Certificate	404
18.1.12. Generating a New Key and Certificate	405
18.1.13. Configure the Firewall for HTTP and HTTPS Using the Command Line	410
18.1.14. Additional Resources	412
CHAPTER 19. MAIL SERVERS	413
19.1. EMAIL PROTOCOLS	413
19.1.1. Mail Transport Protocols	413
19.1.2. Mail Access Protocols	413
19.2. EMAIL PROGRAM CLASSIFICATIONS	416
19.2.1. Mail Transport Agent	416
19.2.2. Mail Delivery Agent	416
19.2.3. Mail User Agent	417
19.3. MAIL TRANSPORT AGENTS	417
19.3.1. Postfix	417
19.3.2. Sendmail	419
19.3.3. Fetchmail	424
19.3.4. Mail Transport Agent (MTA) Configuration	428
19.4. MAIL DELIVERY AGENTS	428
19.4.1. Procmail Configuration	429
19.4.2. Procmail Recipes	430
19.5. MAIL USER AGENTS	435
19.5.1. Securing Communication	435
19.6. ADDITIONAL RESOURCES	436
19.6.1. Installed Documentation	436
19.6.2. Online Documentation	437
19.6.3. Related Books	438
CHAPTER 20. DIRECTORY SERVERS	439
20.1. OPENLDAP	439
20.1.1. Introduction to LDAP	439
20.1.2. Installing the OpenLDAP Suite	441
20.1.3. Configuring an OpenLDAP Server	444
20.1.4. Running an OpenLDAP Server	449
20.1.5. Configuring a System to Authenticate Using OpenLDAP	450
20.1.6. Additional Resources	451
CHAPTER 21. FILE AND PRINT SERVERS	454
21.1. SAMBA	454
21.1.1. Introduction to Samba	454
21.1.2. Samba Daemons and Related Services	454
21.1.3. Connecting to a Samba Share	455
21.1.4. Configuring a Samba Server	458
21.1.5. Starting and Stopping Samba	459
21.1.6. Samba Server Types and the smb.conf File	460
21.1.7. Samba Security Modes	467
21.1.8. Samba Account Information Databases	469
21.1.9. Samba Network Browsing	470
21.1.10. Samba with CUPS Printing Support	471
21.1.11. Samba Distribution Programs	472

21.1.12. Additional Resources	476
21.2. FTP	477
21.2.1. The File Transfer Protocol	477
21.2.2. The vsftpd Server	478
21.2.3. Additional Resources	492
21.3. PRINTER CONFIGURATION	493
21.3.1. Starting the Printer Configuration Tool	494
21.3.2. Starting Printer Setup	494
21.3.3. Adding a Local Printer	495
21.3.4. Adding an AppSocket/HP JetDirect printer	496
21.3.5. Adding an IPP Printer	497
21.3.6. Adding an LPD/LPR Host or Printer	498
21.3.7. Adding a Samba (SMB) printer	499
21.3.8. Selecting the Printer Model and Finishing	501
21.3.9. Printing a Test Page	504
21.3.10. Modifying Existing Printers	504
21.3.11. Additional Resources	510
CHAPTER 22. CONFIGURING NTP USING NTPD	512
22.1. INTRODUCTION TO NTP	512
22.2. NTP STRATA	512
22.3. UNDERSTANDING NTP	513
22.4. UNDERSTANDING THE DRIFT FILE	514
22.5. UTC, TIMEZONES, AND DST	514
22.6. AUTHENTICATION OPTIONS FOR NTP	514
22.7. MANAGING THE TIME ON VIRTUAL MACHINES	515
22.8. UNDERSTANDING LEAP SECONDS	515
22.9. UNDERSTANDING THE NTPD CONFIGURATION FILE	515
22.10. UNDERSTANDING THE NTPD SYSCONFIG FILE	517
22.11. CHECKING IF THE NTP DAEMON IS INSTALLED	517
22.12. INSTALLING THE NTP DAEMON (NTPD)	517
22.13. CHECKING THE STATUS OF NTP	518
22.14. CONFIGURE THE FIREWALL TO ALLOW INCOMING NTP PACKETS	519
22.14.1. Configure the Firewall Using the Graphical Tool	519
22.14.2. Configure the Firewall Using the Command Line	519
22.15. CONFIGURE NTPDATE SERVERS	520
22.16. CONFIGURE NTP	520
22.16.1. Configure Access Control to an NTP Service	521
22.16.2. Configure Rate Limiting Access to an NTP Service	522
22.16.3. Adding a Peer Address	522
22.16.4. Adding a Server Address	522
22.16.5. Adding a Broadcast or Multicast Server Address	523
22.16.6. Adding a Manycast Client Address	523
22.16.7. Adding a Broadcast Client Address	523
22.16.8. Adding a Manycast Server Address	523
22.16.9. Adding a Multicast Client Address	524
22.16.10. Configuring the Burst Option	524
22.16.11. Configuring the iburst Option	524
22.16.12. Configuring Symmetric Authentication Using a Key	525
22.16.13. Configuring the Poll Interval	525
22.16.14. Configuring Server Preference	525
22.16.15. Configuring the Time-to-Live for NTP Packets	525
22.16.16. Configuring the NTP Version to Use	526

22.17. CONFIGURING THE HARDWARE CLOCK UPDATE	526
22.18. CONFIGURING CLOCK SOURCES	526
22.19. ADDITIONAL RESOURCES	526
22.19.1. Installed Documentation	526
22.19.2. Useful Websites	527
CHAPTER 23. CONFIGURING PTP USING PTP4L	528
23.1. INTRODUCTION TO PTP	528
23.1.1. Understanding PTP	528
23.1.2. Advantages of PTP	529
23.2. USING PTP	530
23.2.1. Checking for Driver and Hardware Support	530
23.2.2. Installing PTP	530
23.2.3. Starting ptpt4l	531
23.3. SPECIFYING A CONFIGURATION FILE	532
23.4. USING THE PTP MANAGEMENT CLIENT	533
23.5. SYNCHRONIZING THE CLOCKS	533
23.6. VERIFYING TIME SYNCHRONIZATION	534
23.7. SERVING PTP TIME WITH NTP	536
23.8. SERVING NTP TIME WITH PTP	536
23.9. SYNCHRONIZE TO PTP OR NTP TIME USING TIMEMASTER	537
23.9.1. Starting timemaster as a Service	537
23.9.2. Understanding the timemaster Configuration File	537
23.9.3. Configuring timemaster Options	539
23.10. IMPROVING ACCURACY	540
23.11. ADDITIONAL RESOURCES	541
23.11.1. Installed Documentation	541
23.11.2. Useful Websites	541
PART VII. MONITORING AND AUTOMATION	542
CHAPTER 24. SYSTEM MONITORING TOOLS	543
24.1. VIEWING SYSTEM PROCESSES	543
24.1.1. Using the ps Command	543
24.1.2. Using the top Command	544
24.1.3. Using the System Monitor Tool	545
24.2. VIEWING MEMORY USAGE	547
24.2.1. Using the free Command	547
24.2.2. Using the System Monitor Tool	547
24.3. VIEWING CPU USAGE	548
24.3.1. Using the System Monitor Tool	548
24.4. VIEWING BLOCK DEVICES AND FILE SYSTEMS	549
24.4.1. Using the lsblk Command	549
24.4.2. Using the blkid Command	550
24.4.3. Using the findmnt Command	551
24.4.4. Using the df Command	552
24.4.5. Using the du Command	553
24.4.6. Using the System Monitor Tool	553
24.4.7. Monitoring Files and Directories with gamin	554
24.5. VIEWING HARDWARE INFORMATION	555
24.5.1. Using the lspci Command	555
24.5.2. Using the lsusb Command	556
24.5.3. Using the lspcmcia Command	557
24.5.4. Using the lscpu Command	558

24.6. MONITORING PERFORMANCE WITH NET-SNMP	558
24.6.1. Installing Net-SNMP	558
24.6.2. Running the Net-SNMP Daemon	559
24.6.3. Configuring Net-SNMP	560
24.6.4. Retrieving Performance Data over SNMP	563
24.6.5. Extending Net-SNMP	566
24.7. ADDITIONAL RESOURCES	571
24.7.1. Installed Documentation	571
CHAPTER 25. VIEWING AND MANAGING LOG FILES	573
25.1. INSTALLING RSYSLOG	573
25.1.1. Upgrading to rsyslog version 7	573
25.2. LOCATING LOG FILES	574
25.3. BASIC CONFIGURATION OF RSYSLOG	574
25.3.1. Filters	575
25.3.2. Actions	578
25.3.3. Templates	583
25.3.4. Global Directives	586
25.3.5. Log Rotation	587
25.4. USING THE NEW CONFIGURATION FORMAT	588
25.4.1. Rulesets	588
25.4.2. Compatibility with sysklogd	590
25.5. WORKING WITH QUEUES IN RSYSLOG	590
25.5.1. Defining Queues	591
25.5.2. Creating a New Directory for rsyslog Log Files	594
25.5.3. Managing Queues	595
25.5.4. Using the New Syntax for rsyslog queues	597
25.6. CONFIGURING RSYSLOG ON A LOGGING SERVER	598
25.6.1. Using The New Template Syntax on a Logging Server	601
25.7. USING RSYSLOG MODULES	602
25.7.1. Importing Text Files	603
25.7.2. Exporting Messages to a Database	604
25.7.3. Enabling Encrypted Transport	604
25.7.4. Using RELP	605
25.8. DEBUGGING RSYSLOG	606
25.9. MANAGING LOG FILES IN A GRAPHICAL ENVIRONMENT	606
25.9.1. Viewing Log Files	606
25.9.2. Adding a Log File	609
25.9.3. Monitoring Log Files	610
25.10. ADDITIONAL RESOURCES	611
Installed Documentation	611
Online Documentation	611
See Also	611
CHAPTER 26. UPGRADING MYSQL	613
CHAPTER 27. AUTOMATING SYSTEM TASKS	614
27.1. CRON AND ANACRON	614
27.1.1. Installing Cron and Anacron	614
27.1.2. Running the Crond Service	614
27.1.3. Configuring Anacron Jobs	615
27.1.4. Configuring Cron Jobs	617
27.1.5. Controlling Access to Cron	619
27.1.6. Black and White Listing of Cron Jobs	619

27.2. AT AND BATCH	620
27.2.1. Installing At and Batch	620
27.2.2. Running the At Service	620
27.2.3. Configuring an At Job	621
27.2.4. Configuring a Batch Job	622
27.2.5. Viewing Pending Jobs	622
27.2.6. Additional Command-Line Options	622
27.2.7. Controlling Access to At and Batch	623
27.3. ADDITIONAL RESOURCES	623
CHAPTER 28. AUTOMATIC BUG REPORTING TOOL (ABRT)	624
28.1. INSTALLING ABRT AND STARTING ITS SERVICES	625
28.2. USING THE GRAPHICAL USER INTERFACE	627
28.3. USING THE COMMAND-LINE INTERFACE	634
28.3.1. Viewing Problems	634
28.3.2. Reporting Problems	636
28.3.3. Deleting Problems	637
28.4. CONFIGURING ABRT	638
28.4.1. ABRT Events	638
28.4.2. Standard ABRT Installation Supported Events	640
28.4.3. Event Configuration in ABRT GUI	641
28.4.4. ABRT Specific Configuration	643
28.4.5. Configuring ABRT to Detect a Kernel Panic	645
28.4.6. Automatic Downloads and Installation of Debuginfo Packages	646
28.4.7. Configuring Automatic Reporting for Specific Types of Crashes	646
28.4.8. Uploading and Reporting Using a Proxy Server	647
28.4.9. Configuring Automatic Reporting	647
28.5. CONFIGURING CENTRALIZED CRASH COLLECTION	649
28.5.1. Configuration Steps Required on a Dedicated System	649
28.5.2. Configuration Steps Required on a Client System	650
28.5.3. Saving Package Information	651
28.5.4. Testing ABRT's Crash Detection	652
CHAPTER 29. OPROFILE	653
29.1. OVERVIEW OF TOOLS	653
29.2. CONFIGURING OPROFILE	654
29.2.1. Specifying the Kernel	654
29.2.2. Setting Events to Monitor	655
29.2.3. Separating Kernel and User-space Profiles	658
29.3. STARTING AND STOPPING OPROFILE	659
29.4. SAVING DATA	659
29.5. ANALYZING THE DATA	660
29.5.1. Using oreport	661
29.5.2. Using oreport on a Single Executable	661
29.5.3. Getting more detailed output on the modules	663
29.5.4. Using oannotate	664
29.6. UNDERSTANDING /DEV/OPROFILE/	664
29.7. EXAMPLE USAGE	665
29.8. OPROFILE SUPPORT FOR JAVA	665
29.8.1. Profiling Java Code	665
29.9. GRAPHICAL INTERFACE	665
29.10. OPROFILE AND SYSTEMTAP	669
29.11. ADDITIONAL RESOURCES	669

29.11.1. Installed Docs	669
29.11.2. Useful Websites	669
PART VIII. KERNEL, MODULE AND DRIVER CONFIGURATION	670
CHAPTER 30. MANUALLY UPGRADING THE KERNEL	671
30.1. OVERVIEW OF KERNEL PACKAGES	671
30.2. PREPARING TO UPGRADE	672
30.3. DOWNLOADING THE UPGRADED KERNEL	673
30.4. PERFORMING THE UPGRADE	674
30.5. VERIFYING THE INITIAL RAM DISK IMAGE Verifying the Initial RAM Disk Image and Kernel on IBM eServer System i	674 676
30.6. VERIFYING THE BOOT LOADER 30.6.1. Configuring the GRUB Boot Loader	676 677
30.6.2. Configuring the Loopback Device Limit	678
30.6.3. Configuring the OS/400 Boot Loader	679
30.6.4. Configuring the YABOOT Boot Loader	679
CHAPTER 31. WORKING WITH KERNEL MODULES	681
31.1. LISTING CURRENTLY-LOADED MODULES	681
31.2. DISPLAYING INFORMATION ABOUT A MODULE	682
31.3. LOADING A MODULE	684
31.4. UNLOADING A MODULE	685
31.5. BLACKLISTING A MODULE	686
31.6. SETTING MODULE PARAMETERS 31.6.1. Loading a Customized Module - Temporary Changes	688 688
31.6.2. Loading a Customized Module - Persistent Changes	689
31.7. PERSISTENT MODULE LOADING	690
31.8. SPECIFIC KERNEL MODULE CAPABILITIES 31.8.1. Using Channel Bonding	691 691
31.9. ADDITIONAL RESOURCES Installed Documentation	697 697
Installable Documentation	697
Online Documentation	697
CHAPTER 32. THE KDUMP CRASH RECOVERY SERVICE	698
32.1. INSTALLING THE KDUMP SERVICE	698
32.2. CONFIGURING THE KDUMP SERVICE 32.2.1. Configuring kdump at First Boot	698 698
32.2.2. Using the Kernel Dump Configuration Utility	699
32.2.3. Configuring kdump on the Command Line	705
32.2.4. Testing the Configuration	709
32.3. ANALYZING THE CORE DUMP 32.3.1. Running the crash Utility	709 710
32.3.2. Displaying the Message Buffer	711
32.3.3. Displaying a Backtrace	712
32.3.4. Displaying a Process Status	712
32.3.5. Displaying Virtual Memory Information	713
32.3.6. Displaying Open Files	714
32.3.7. Exiting the Utility	714
32.4. USING FADUMP ON IBM POWERPC HARDWARE Enabling fadump	714 715
32.5. USING SADUMP ON FUJITSU PRIMEQUEST SYSTEMS How to use sadump	715 716

32.5.1. Configure Red Hat Enterprise Linux for sadump	716
32.5.2. Check the memory dump	717
32.6. ADDITIONAL RESOURCES	717
Installed Documentation	717
Useful Websites	717
PART IX. SYSTEM RECOVERY	719
CHAPTER 33. SYSTEM RECOVERY	720
33.1. RESCUE MODE	720
33.2. SINGLE-USER MODE	722
33.3. EMERGENCY MODE	723
33.4. RESOLVING PROBLEMS IN SYSTEM RECOVERY MODES	723
CHAPTER 34. RELAX-AND-RECOVER (REAR)	726
34.1. BASIC REAR USAGE	726
34.1.1. Installing ReaR	726
34.1.2. Configuring ReaR	726
34.1.3. Creating a Rescue System	727
34.1.4. Scheduling ReaR	727
34.1.5. Performing a System Rescue	728
34.2. INTEGRATING REAR WITH BACKUP SOFTWARE	731
34.2.1. The Built-in Backup Method	731
34.2.2. Supported Backup Methods	733
34.2.3. Unsupported Backup Methods	733
APPENDIX A. CONSISTENT NETWORK DEVICE NAMING	735
A.1. AFFECTED SYSTEMS	735
A.2. SYSTEM REQUIREMENTS	736
A.3. ENABLING AND DISABLING THE FEATURE	736
A.4. NOTES FOR ADMINISTRATORS	736
APPENDIX B. RPM	738
B.1. RPM DESIGN GOALS	739
B.2. USING RPM	739
B.2.1. Finding RPM Packages	740
B.2.2. Installing and Upgrading	740
B.2.3. Configuration File Changes	743
B.2.4. Uninstalling	743
B.2.5. Freshening	744
B.2.6. Querying	745
B.2.7. Verifying	746
B.3. CHECKING A PACKAGE'S SIGNATURE	746
B.3.1. Importing Keys	747
B.3.2. Verifying Signature of Packages	747
B.4. PRACTICAL AND COMMON EXAMPLES OF RPM USAGE	748
B.5. ADDITIONAL RESOURCES	749
B.5.1. Installed Documentation	749
B.5.2. Useful Websites	750
APPENDIX C. THE X WINDOW SYSTEM	751
C.1. THE X SERVER	751
C.2. DESKTOP ENVIRONMENTS AND WINDOW MANAGERS	751
C.2.1. Maximum number of concurrent GUI sessions	752
C.2.2. Desktop Environments	752

C.2.3. Window Managers	752
C.3. X SERVER CONFIGURATION FILES	753
C.3.1. The Structure of the Configuration	753
C.3.2. The xorg.conf.d Directory	754
C.3.3. The xorg.conf File	754
C.4. FONTS	761
C.4.1. Adding Fonts to Fontconfig	762
C.5. RUNLEVELS AND X	762
C.5.1. Runlevel 3	762
C.5.2. Runlevel 5	763
C.6. ACCESSING GRAPHICAL APPLICATIONS REMOTELY	764
C.7. ADDITIONAL RESOURCES	764
C.7.1. Installed Documentation	764
C.7.2. Useful Websites	764
APPENDIX D. THE SYSCONFIG DIRECTORY	766
D.1. FILES IN THE /ETC/SYSCONFIG/ DIRECTORY	766
D.1.1. /etc/sysconfig/arpwatch	766
D.1.2. /etc/sysconfig/authconfig	766
D.1.3. /etc/sysconfig/autofs	769
D.1.4. /etc/sysconfig/clock	771
D.1.5. /etc/sysconfig/dhcpd	771
D.1.6. /etc/sysconfig/firstboot	771
D.1.7. /etc/sysconfig/i18n	772
D.1.8. /etc/sysconfig/init	772
D.1.9. /etc/sysconfig/ip6tables-config	774
D.1.10. /etc/sysconfig/kernel	775
D.1.11. /etc/sysconfig/keyboard	775
D.1.12. /etc/sysconfig/ldap	776
D.1.13. /etc/sysconfig/named	777
D.1.14. /etc/sysconfig/network	778
D.1.15. /etc/sysconfig/ntpd	779
D.1.16. /etc/sysconfig/quagga	779
D.1.17. /etc/sysconfig/radvd	780
D.1.18. /etc/sysconfig/samba	780
D.1.19. /etc/sysconfig/saslauthd	781
D.1.20. /etc/sysconfig/selinux	781
D.1.21. /etc/sysconfig/sendmail	782
D.1.22. /etc/sysconfig/spamassassin	782
D.1.23. /etc/sysconfig/squid	782
D.1.24. /etc/sysconfig/system-config-users	783
D.1.25. /etc/sysconfig/vncservers	783
D.1.26. /etc/sysconfig/xinetd	784
D.2. DIRECTORIES IN THE /ETC/SYSCONFIG/ DIRECTORY	784
D.3. ADDITIONAL RESOURCES	785
D.3.1. Installed Documentation	785
APPENDIX E. THE PROC FILE SYSTEM	786
E.1. A VIRTUAL FILE SYSTEM	786
E.1.1. Viewing Virtual Files	786
E.1.2. Changing Virtual Files	787
E.2. TOP-LEVEL FILES WITHIN THE PROC FILE SYSTEM	787
E.2.1. /proc/buddyinfo	787

E.2.2. /proc/cmdline	788
E.2.3. /proc/cpuinfo	788
E.2.4. /proc/crypto	789
E.2.5. /proc/devices	790
E.2.6. /proc/dma	790
E.2.7. /proc/execdomains	791
E.2.8. /proc/fb	791
E.2.9. /proc/filesystems	791
E.2.10. /proc/interrupts	792
E.2.11. /proc/iomem	792
E.2.12. /proc/ioports	793
E.2.13. /proc/kcore	793
E.2.14. /proc/kmsg	794
E.2.15. /proc/loadavg	794
E.2.16. /proc/locks	794
E.2.17. /proc/mdstat	795
E.2.18. /proc/meminfo	795
E.2.19. /proc/misc	798
E.2.20. /proc/modules	799
E.2.21. /proc/mounts	799
E.2.22. /proc/mtrr	800
E.2.23. /proc/partitions	800
E.2.24. /proc/slabinfo	801
E.2.25. /proc/stat	802
E.2.26. /proc/swaps	803
E.2.27. /proc/sysrq-trigger	803
E.2.28. /proc/uptime	803
E.2.29. /proc/version	803
E.3. DIRECTORIES WITHIN /PROC/	803
E.3.1. Process Directories	804
E.3.2. /proc/bus/	806
E.3.3. /proc/bus/pci	806
E.3.4. /proc/driver/	807
E.3.5. /proc/fs	808
E.3.6. /proc/irq/	808
E.3.7. /proc/net/	808
E.3.8. /proc/scsi/	809
E.3.9. /proc/sys/	811
E.3.10. /proc/sysvipc/	820
E.3.11. /proc/tty/	820
E.3.12. /proc/PID/	821
E.4. USING THE SYSCTL COMMAND	822
E.5. ADDITIONAL RESOURCES	823
Installable Documentation	823
APPENDIX F. REVISION HISTORY	824
INDEX	826

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. KEYBOARD CONFIGURATION

This chapter describes how to change the keyboard layout, as well as how to add the **Keyboard Indicator** applet to the panel. It also covers the option to enforce a typing break, and explains both advantages and disadvantages of doing so.

1.1. CHANGING THE KEYBOARD LAYOUT

The installation program has allowed you to configure a keyboard layout for your system. However, the default settings may not always suit your current needs. To configure a different keyboard layout after the installation, use the **Keyboard Preferences** tool.

To open **Keyboard Layout Preferences**, select **System → Preferences → Keyboard** from the panel, and click the **Layouts** tab.

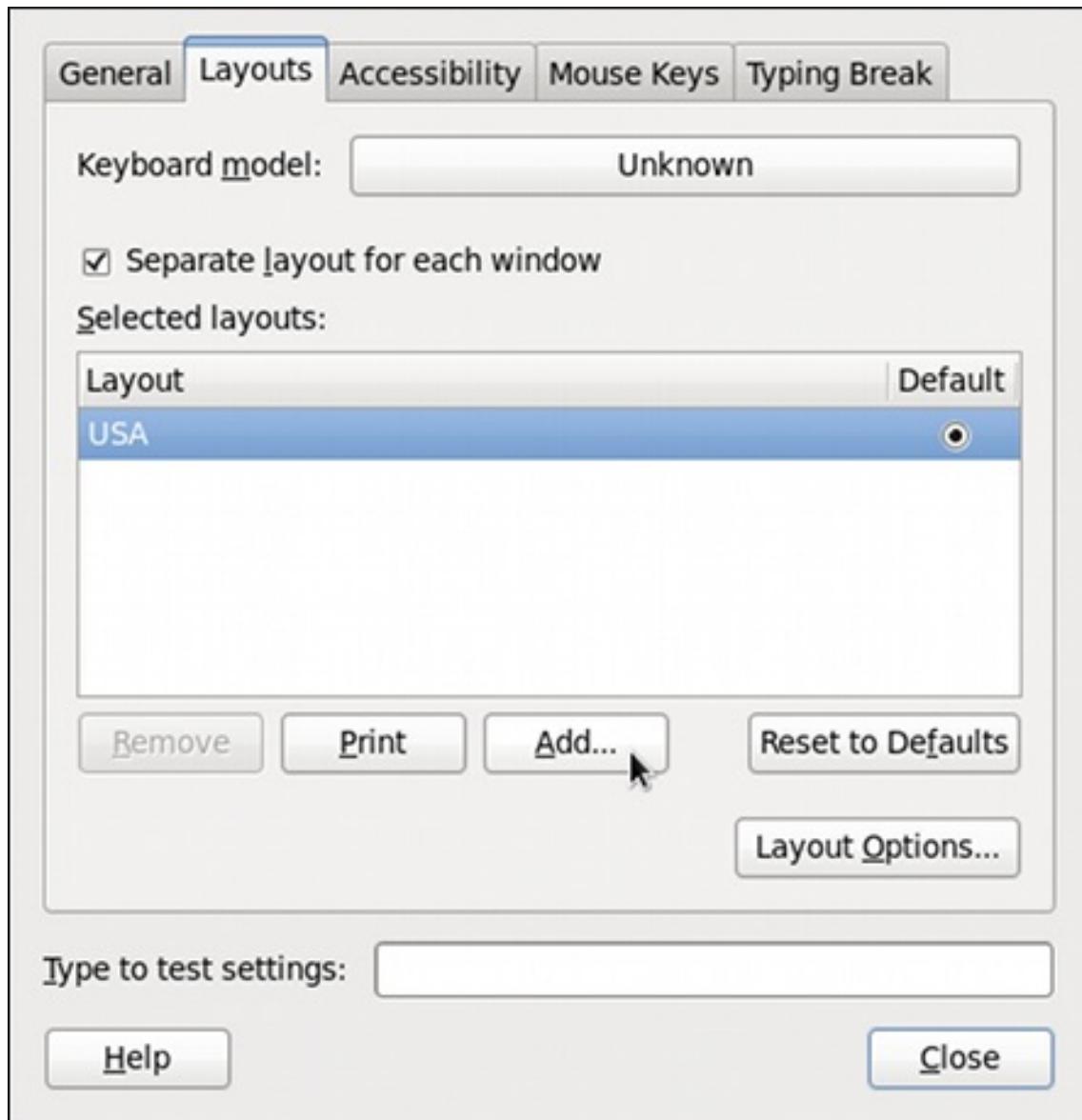


Figure 1.1. Keyboard Layout Preferences

You will be presented with a list of available layouts. To add a new one, click the **Add** button below the list, and you will be prompted to choose which layout you want to add.



Figure 1.2. Choosing a layout

Currently, there are two ways how to choose the keyboard layout: you can either find it by the country it is associated with (the **By country** tab), or you can select it by language (the **By language** tab). In either case, first select the desired country or language from the **Country** or **Language** pulldown menu, then specify the variant from the **Variants** menu. The preview of the layout changes immediately. To confirm the selection, click **Add**.



Figure 1.3. Selecting the default layout

The layout should appear in the list. To make it the default, select the radio button next to its name. The changes take effect immediately. Note that there is a text-entry field at the bottom of the window where you can safely test your settings. Once you are satisfied, click **Close** to close the window.

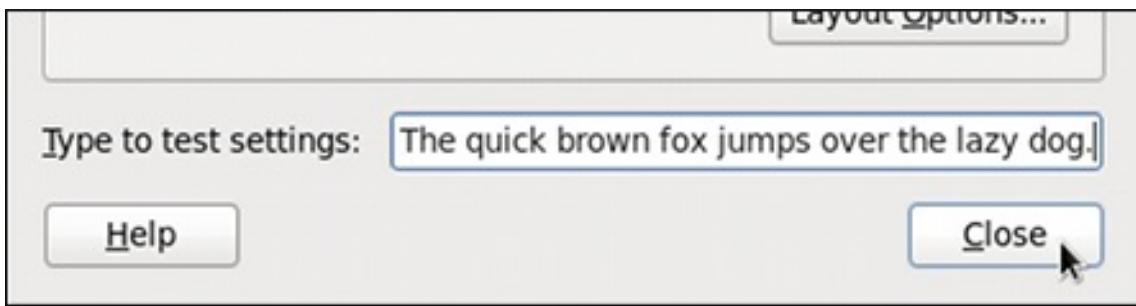
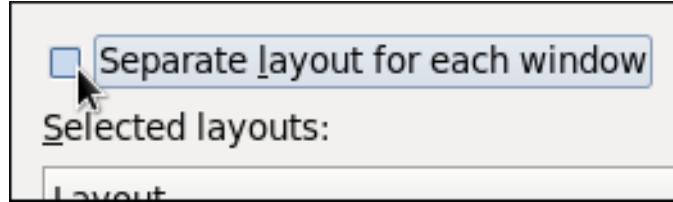


Figure 1.4. Testing the layout



NOTE

By default, changing the keyboard layout affects the active window only. This means that if you change the layout and switch to another window, this window will use the old one, which might be confusing. To turn this behavior off, clear the **Separate layout for each window** check box.



Doing this has its drawbacks though, as you will no longer be able to choose the default layout by selecting the radio button as shown in [Figure 1.3, "Selecting the default layout"](#). To make the layout the default, drag it to the beginning of the list.



1.2. ADDING THE KEYBOARD LAYOUT INDICATOR

If you want to see what keyboard layout you are currently using, or you would like to switch between different layouts with a single mouse click, add the **Keyboard Indicator** applet to the panel. To do so, right-click the empty space on the main panel, and select the **Add to Panel** option from the pulldown menu.

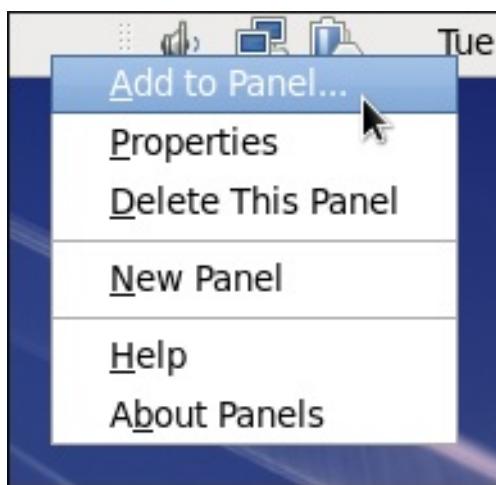


Figure 1.5. Adding a new applet

You will be presented with a list of available applets. Scroll through the list (or start typing “keyboard” into the search field at the top of the window), select **Keyboard Indicator**, and click the **Add** button.

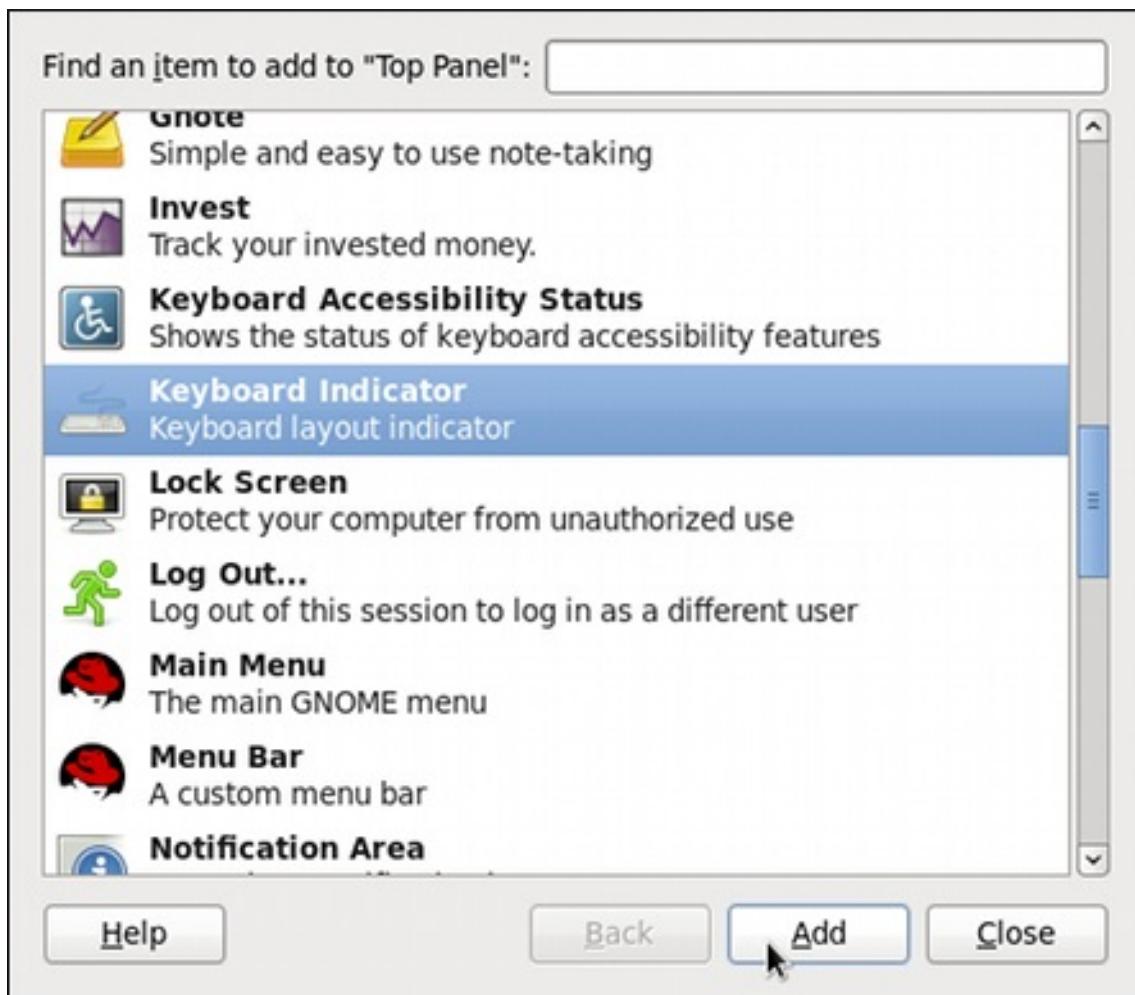


Figure 1.6. Selecting the Keyboard Indicator

The applet appears immediately, displaying the shortened name of the country the current layout is associated with. To display the actual variant, hover the pointer over the applet icon.



Figure 1.7. The Keyboard Indicator applet

1.3. SETTING UP A TYPING BREAK

Typing for a long period of time can be not only tiring, but it can also increase the risk of serious health problems, such as carpal tunnel syndrome. One way of preventing this is to configure the system to enforce typing breaks. To do so, select **System → Preferences → Keyboard** from the panel, click the **Typing Break** tab, and select the **Lock screen to enforce typing break** check box.

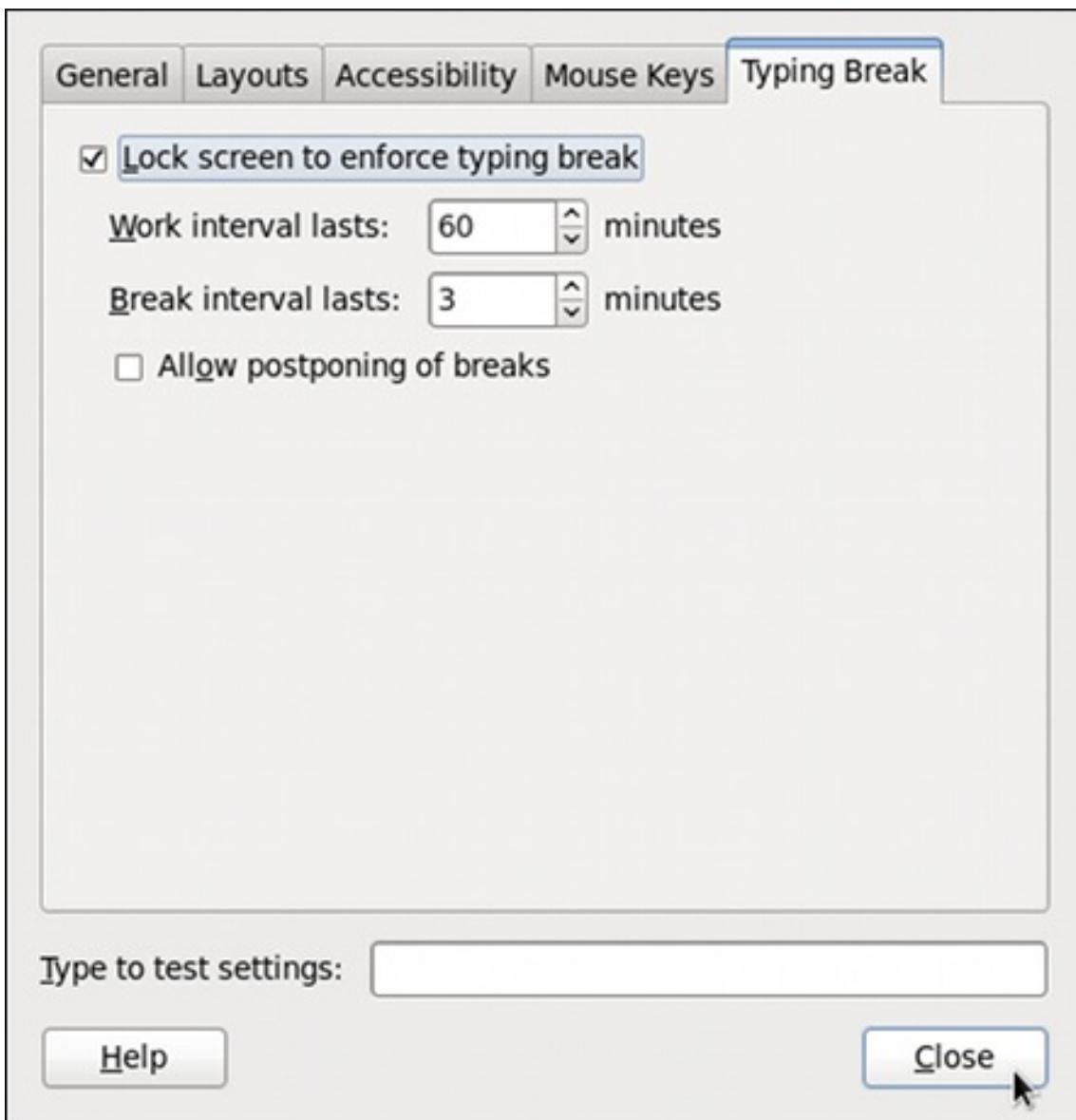


Figure 1.8. Typing Break Properties

To increase or decrease the allowed typing time before the break is enforced, click the up or down button next to the **Work interval lasts** label respectively. You can do the same with the **Break interval lasts** setting to alter the length of the break itself. Finally, select the **Allow postponing of breaks** check box if you want to be able to delay the break in case you need to finish the work. The changes take effect immediately.

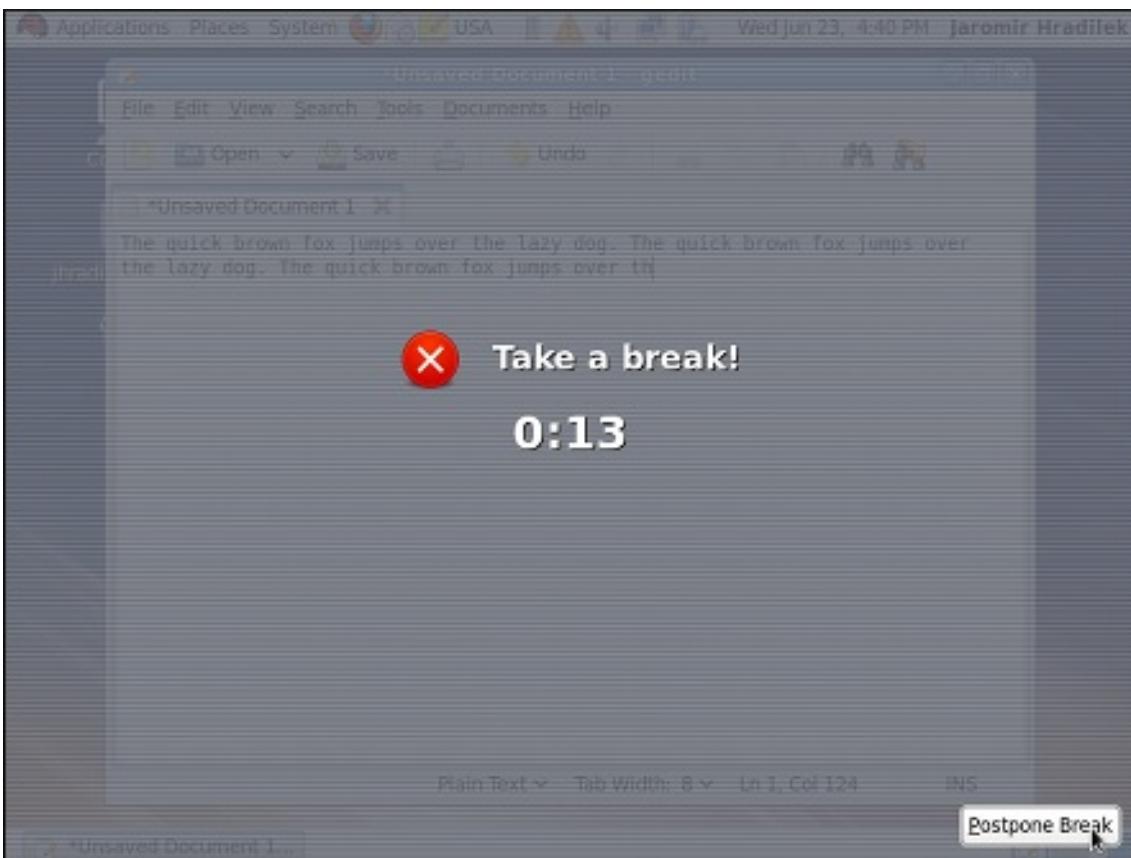


Figure 1.9. Taking a break

Next time you reach the time limit, you will be presented with a screen advising you to take a break, and a clock displaying the remaining time. If you have enabled it, the **Postpone Break** button will be located at the bottom right corner of the screen.

CHAPTER 2. DATE AND TIME CONFIGURATION

This chapter covers setting the system date and time in Red Hat Enterprise Linux, both manually and using the Network Time Protocol (NTP), as well as setting the adequate time zone. Two methods are covered: setting the date and time using the **Date/Time Properties** tool, and doing so on the command line.

2.1. DATE/TIME PROPERTIES TOOL

The **Date/Time Properties** tool allows the user to change the system date and time, to configure the time zone used by the system, and to set up the Network Time Protocol daemon to synchronize the system clock with a time server. Note that to use this application, you must be running the X Window System (see [Appendix C, The X Window System](#) for more information on this topic).

To start the tool, select **System → Administration → Date & Time** from the panel, or type the **system-config-date** command at a shell prompt (e.g., *xterm* or *GNOME Terminal*). Unless you are already authenticated, you will be prompted to enter the superuser password.

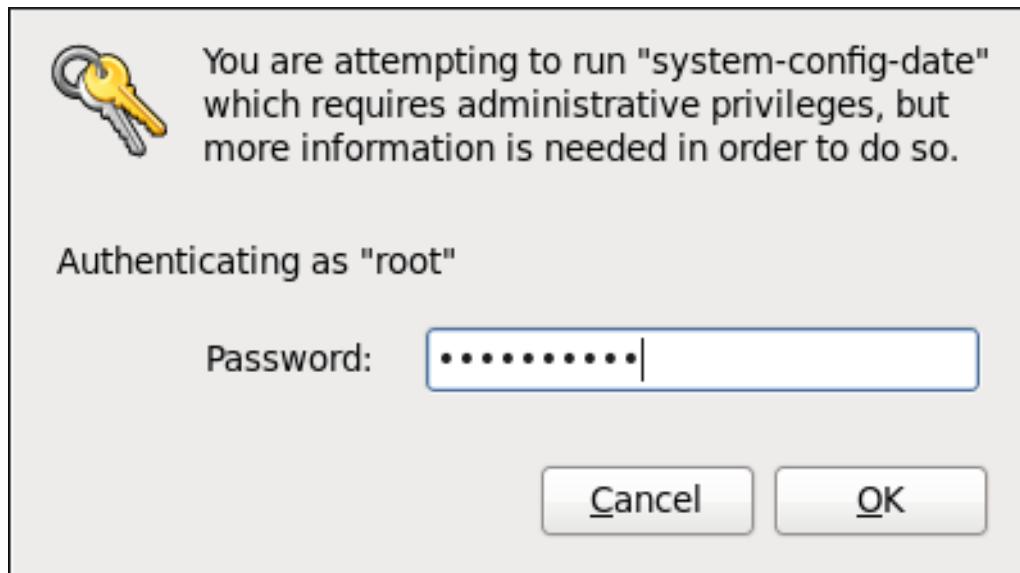


Figure 2.1. Authentication Query

2.1.1. Date and Time Properties

As shown in [Figure 2.2, “Date and Time Properties”](#), the **Date/Time Properties** tool is divided into two separate tabs. The tab containing the configuration of the current date and time is shown by default.

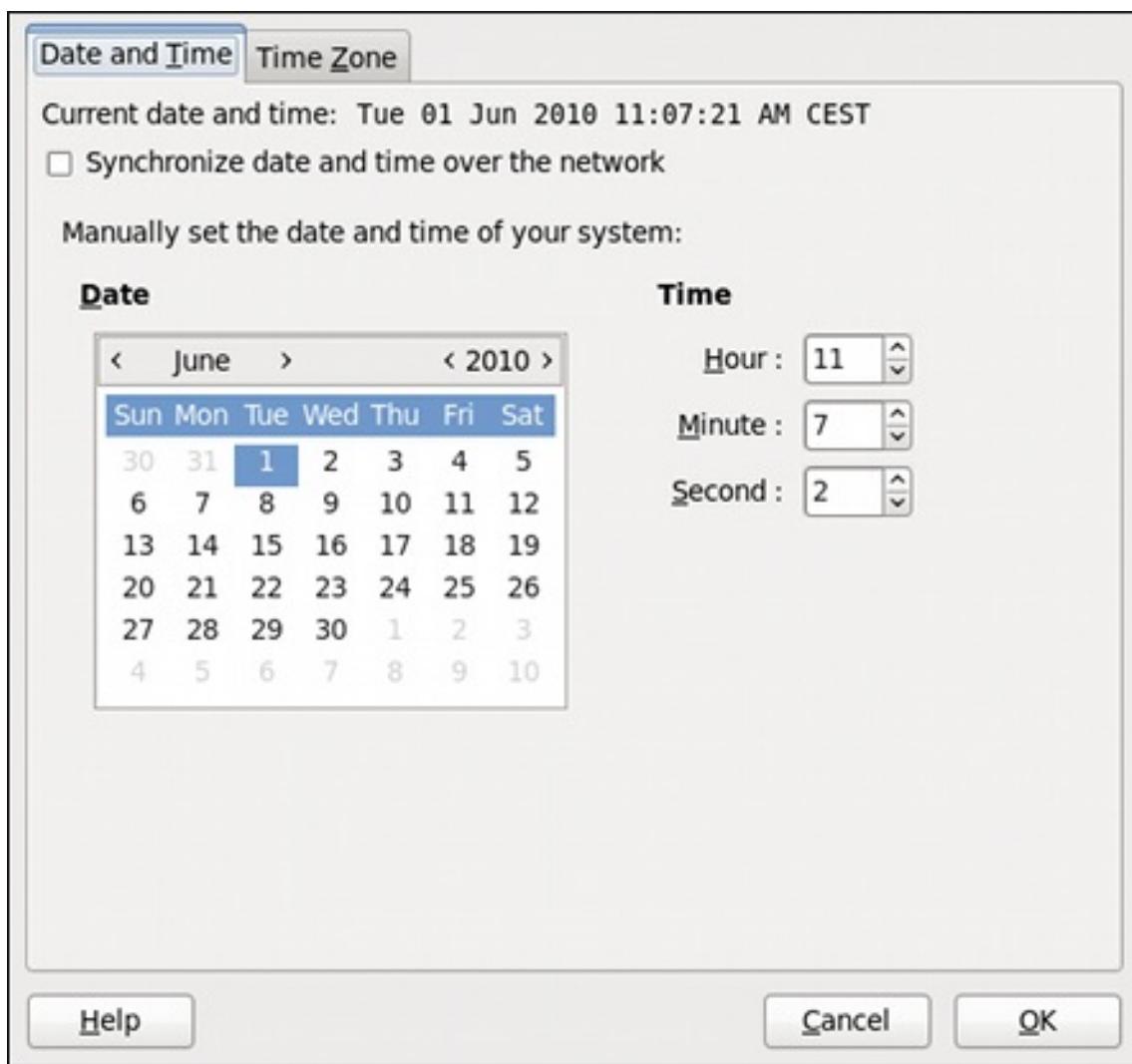


Figure 2.2. Date and Time Properties

To set up your system manually, follow these steps:

1. *Change the current date.* Use the arrows to the left and right of the month and year respectively. Then click inside the calendar to select the day of the month.
2. *Change the current time.* Use the up and down arrow buttons beside the **Hour**, **Minute**, and **Second**, or replace the values directly.

Click the **OK** button to apply the changes and exit the application.

2.1.2. Network Time Protocol Properties

If you prefer an automatic setup, select the check box labeled **Synchronize date and time over the network** instead. This will display the list of available NTP servers as shown in [Figure 2.3, "Network Time Protocol Properties"](#).

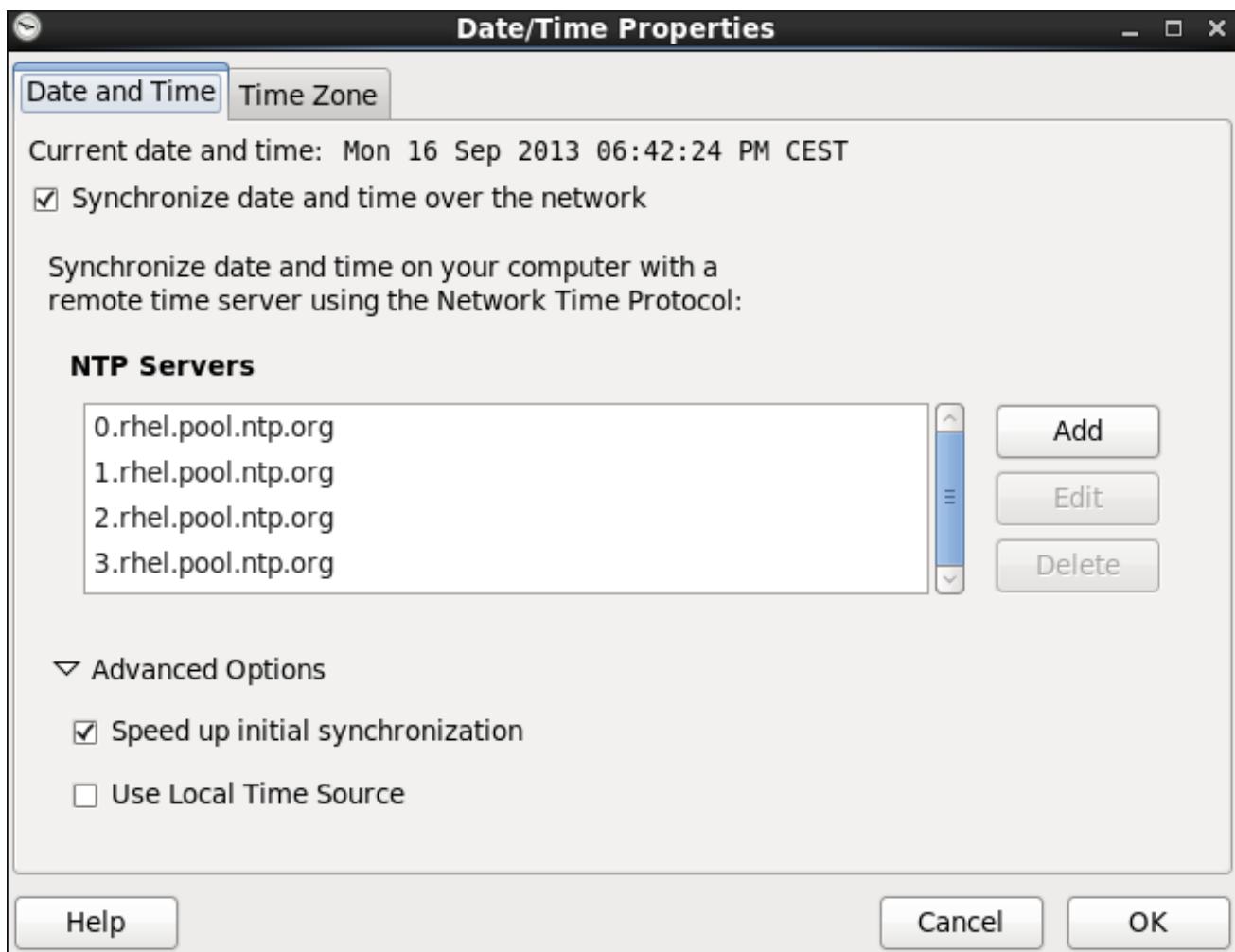


Figure 2.3. Network Time Protocol Properties

Here you can choose one of the predefined servers, edit a predefined server by clicking the **Edit** button, or add a new server name by clicking **Add**. In the **Advanced Options**, you can also select whether you want to speed up the initial synchronization of the system clock, or if you want to use a local time source.



NOTE

Your system does not start synchronizing with the NTP server until you click the **OK** button at the bottom of the window to confirm your changes.

Click the **OK** button to apply any changes made to the date and time settings and exit the application.

2.1.3. Time Zone Properties

To configure the system time zone, click the **Time Zone** tab as shown in [Figure 2.4, "Time Zone Properties"](#).



Figure 2.4. Time Zone Properties

There are two common approaches to the time zone selection:

1. *Using the interactive map.* Click “zoom in” and “zoom out” buttons next to the map, or click on the map itself to zoom into the selected region. Then choose the city specific to your time zone. A red **X** appears and the time zone selection changes in the list below the map.
2. *Use the list below the map.* To make the selection easier, cities and countries are grouped within their specific continents. Note that non-geographic time zones have also been added to address needs in the scientific community.

If your system clock is set to use UTC, select the **System clock uses UTC** option. UTC stands for the *Universal Time, Coordinated*, also known as *Greenwich Mean Time* (GMT). Other time zones are determined by adding or subtracting from the UTC time.

Click **OK** to apply the changes and exit the program.

2.2. COMMAND LINE CONFIGURATION

In case your system does not have the **Date/Time Properties** tool installed, or the X Window Server is not running, you will have to change the system date and time on the command line. Note that in order to perform actions described in this section, you have to be logged in as a superuser:

```
~]$ su -
Password:
```

2.2.1. Date and Time Setup

The **date** command allows the superuser to set the system date and time manually:

1. *Change the current date.* Type the command in the following form at a shell prompt, replacing the *YYYY* with a four-digit year, *MM* with a two-digit month, and *DD* with a two-digit day of the month:

```
~]# date +%D -s YYYY-MM-DD
```

For example, to set the date to 2 June 2010, type:

```
~]# date +%D -s 2010-06-02
```

2. *Change the current time.* Use the following command, where *HH* stands for an hour, *MM* is a minute, and *SS* is a second, all typed in a two-digit form:

```
~]# date +%T -s HH:MM:SS
```

If your system clock is set to use UTC (Coordinated Universal Time), add the following option:

```
~]# date +%T -s HH:MM:SS -u
```

For instance, to set the system clock to 11:26 PM using the UTC, type:

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

You can check your current settings by typing **date** without any additional argument:

Example 2.1. Displaying the current date and time

```
~]$ date
Wed Jun 2 11:58:48 CEST 2010
```

2.2.2. Network Time Protocol Setup

As opposed to the manual setup described above, you can also synchronize the system clock with a remote server over the Network Time Protocol (NTP). For the one-time synchronization only, use the **ntpdate** command:

1. Firstly, check whether the selected NTP server is accessible:

```
~]# ntpdate -q server_address
```

For example:

```
~]# ntpdate -q 0.rhel.pool.ntp.org
```

2. When you find a satisfactory server, run the **ntpdate** command followed by one or more server addresses:

```
~]# ntpdate server_address...
```

For instance:

```
~]# ntpdate 0.rhel.pool.ntp.org 1.rhel.pool.ntp.org
```

Unless an error message is displayed, the system time should now be set. You can check the current by setting typing **date** without any additional arguments as shown in [Section 2.2.1, “Date and Time Setup”](#).

3. In most cases, these steps are sufficient. Only if you really need one or more system services to always use the correct time, enable running the **ntpdate** at boot time:

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

For more information about system services and their setup, see [Chapter 12, Services and Daemons](#).



NOTE

If the synchronization with the time server at boot time keeps failing, i.e., you find a relevant error message in the **/var/log/boot.log** system log, try to add the following line to **/etc/sysconfig/network**:

```
NETWORKWAIT=1
```

However, the more convenient way is to set the **ntpd** daemon to synchronize the time at boot time automatically:

1. Open the NTP configuration file **/etc/ntp.conf** in a text editor such as **vi** or **nano**, or create a new one if it does not already exist:

```
~]# nano /etc/ntp.conf
```

2. Now add or edit the list of public NTP servers. If you are using Red Hat Enterprise Linux 6, the file should already contain the following lines, but feel free to change or expand these according to your needs:

```
server 0.rhel.pool.ntp.org iburst
server 1.rhel.pool.ntp.org iburst
server 2.rhel.pool.ntp.org iburst
server 3.rhel.pool.ntp.org iburst
```

The **iburst** directive at the end of each line is to speed up the initial synchronization. As of Red Hat Enterprise Linux 6.5 it is added by default. If upgrading from a previous minor release, and your **/etc/ntp.conf** file has been modified, then the upgrade to Red Hat Enterprise Linux 6.5 will create a new file **/etc/ntp.conf.rpmnew** and will not alter the existing **/etc/ntp.conf** file.

3. Once you have the list of servers complete, in the same file, set the proper permissions, giving the unrestricted access to localhost only:

```
restrict default kod nomodify notrap nopeer noquery  
restrict -6 default kod nomodify notrap nopeer noquery  
restrict 127.0.0.1  
restrict -6 ::1
```

4. Save all changes, exit the editor, and restart the NTP daemon:

```
~]# service ntpd restart
```

5. Make sure that **ntpd** is started at boot time:

```
~]# chkconfig ntpd on
```

CHAPTER 3. MANAGING USERS AND GROUPS

3.1. WHAT USERS AND GROUPS ARE

The control of users and groups is a core element of Red Hat Enterprise Linux system administration. The user of the system is either a human being or an account used by specific applications identified by a unique numerical identification number called *user ID* (UID). Users within a group can have read permissions, write permissions, execute permissions or any combination of read/write/execute permissions for files owned by that group.

Red Hat Enterprise Linux supports access *control lists* (ACLs) for files and directories which allow permissions for specific users outside of the owner to be set. For more information about this feature, see the *Access Control Lists* chapter of the *Red Hat Enterprise Linux 6 Storage Administration Guide*.

A group is an organization unit tying users together for a common purpose, which can be reading permissions, writing permission, or executing permission for files owned by that group. Similar to UID, each group is associated with a group ID (GID).

NOTE

Red Hat Enterprise Linux reserves user and group IDs below 500 for system users and groups. By default, the **User Manager** does not display the system users. Reserved user and group IDs are documented in the setup package. To view the documentation, use this command:

```
cat /usr/share/doc/setup-2.8.14/uidgid
```

The recommended practice is to assign non-reserved IDs starting at 5,000, as the reserved range can increase in the future. To make the IDs assigned to new users by default start at 5,000, change the **UID_MIN** and **GID_MIN** directives in the **/etc/login.defs** file:

```
[file contents truncated]
UID_MIN      5000
[file contents truncated]
GID_MIN      5000
[file contents truncated]
```

Even with new user and group IDs beginning with 5,000, it is recommended not to raise IDs reserved by system above 500 to avoid conflict with systems that retain the 500 limit.

Each user is a member of exactly one primary group and zero or more supplementary groups. By default, when a file is created, the file's owner is its creator and the file's group is the creator's primary group. A user can temporarily change what group is their primary group with the **newgrp** command, after which all newly created files are owned by the new group. A supplementary group serves to grant a certain set of users, its members, access to a certain set of files, those owned by this group.

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.

By default, a file or directory can be modified only by its creator. The setting that determines what permissions are applied to a newly created file or directory is called a **umask** and can be configured in

the **/etc/bashrc** file for all users, or in **~/.bashrc** for each user individually. The configuration in **~/.bashrc** overrides the configuration in **/etc/bashrc**. Additionally, the **umask** command overrides the default permissions for the duration of the shell session.

To authenticate, a user enters their password. A hash sum is generated from the entered string and compared to the hash sum of the user's password. If the hash sums match, the user authenticates successfully.

Hash sums of user passwords are stored in the **/etc/shadow** file, which is only readable by the **root** user. The file also stores information about password aging and policies for specific accounts. The default values for a newly created account are stored in the **/etc/login.defs** and **/etc/default/useradd** files. The *Red Hat Enterprise Linux 6 Security Guide* provides more security-related information about users and groups.

3.2. MANAGING USERS VIA THE USER MANAGER APPLICATION

The **User Manager** application allows you to view, modify, add, and delete local users and groups in the graphical user interface.

To start the User Manager application:

- From the toolbar, select **System → Administration → Users and Groups**.
- Or, type **system-config-users** at the shell prompt.



NOTE

Unless you have superuser privileges, the application will prompt you to authenticate as **root**.

3.2.1. Viewing Users

In order to display the main window of the **User Manager** to view users, from the toolbar of **User Manager** select **Edit → Preferences**. If you want to view *all* the users, that is including system users, clear the **Hide system users and groups** check box.

The **Users** tab provides a list of local users along with additional information about their user ID, primary group, home directory, login shell, and full name.

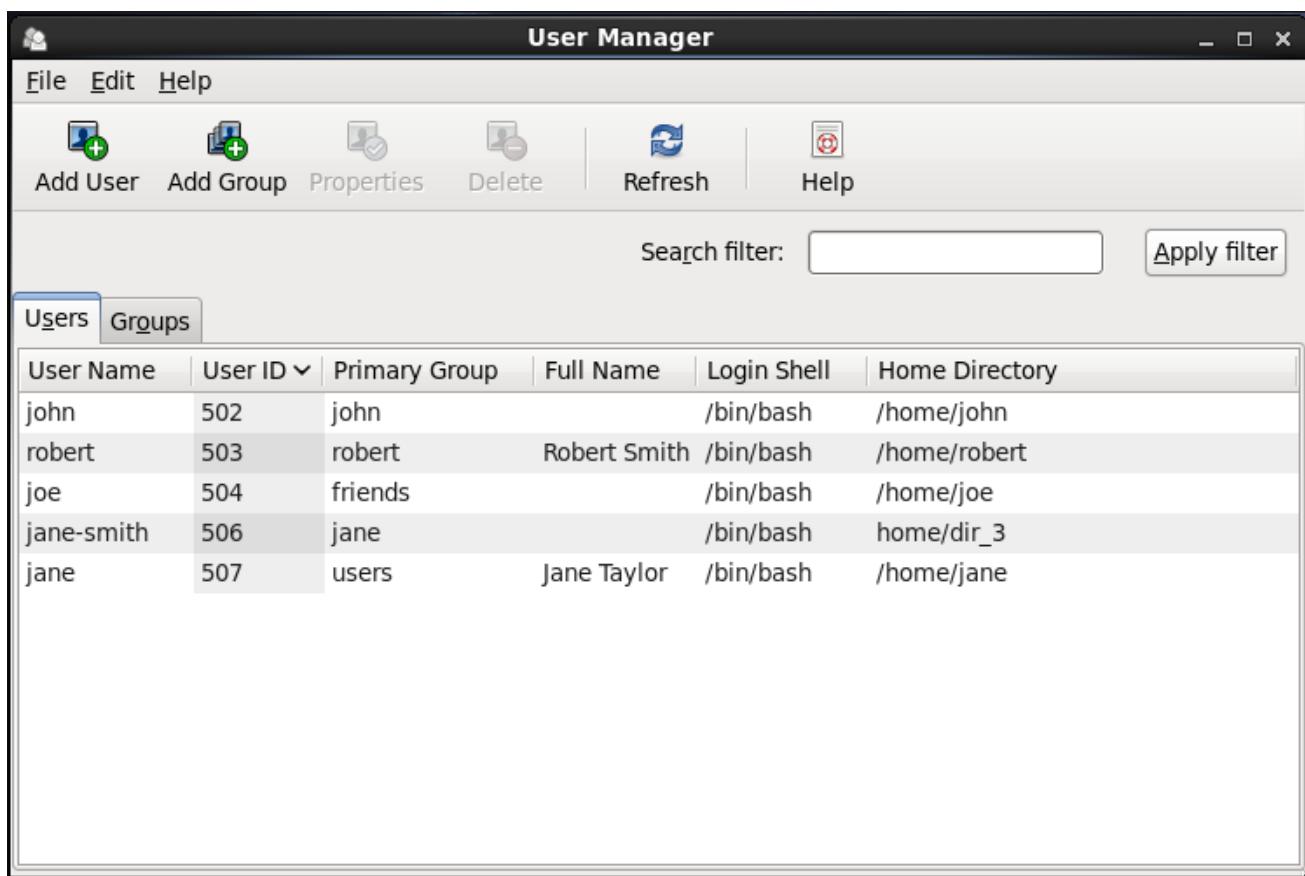


Figure 3.1. Viewing Users

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

3.2.2. Adding a New User

If there is a new user you need to add to the system, follow this procedure:

1. Click the **Add User** button.
2. Enter the user name and full name in the appropriate fields
3. Type the user's password in the **Password** and **Confirm Password** fields. The password must be at least six characters long.



NOTE

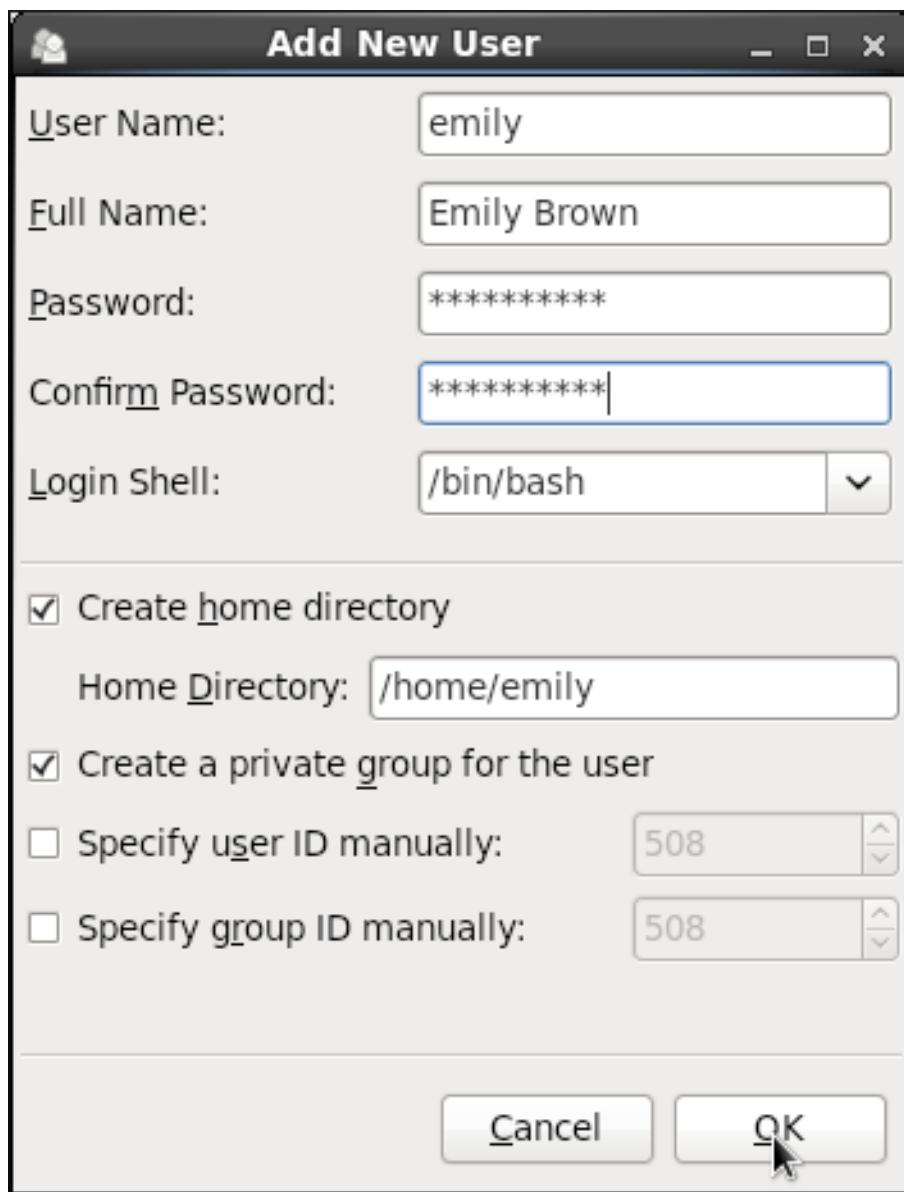
For safety reasons, choose a long password not based on a dictionary term; use a combination of letters, numbers, and special characters.

4. Select a login shell for the user from the **Login Shell** drop-down list or accept the default value of **/bin/bash**.
5. Clear the **Create home directory** check box if you choose not to create the home directory for a new user in **/home/username**.

You can also change this home directory by editing the content of the **Home Directory** text box. Note that when the home directory is created, default configuration files are copied into it from the **/etc/skel/** directory.

6. Clear the **Create a private group for the user** check box if you do not want a unique group with the same name as the user to be created. User private group (UPG) is a group assigned to a user account to which that user exclusively belongs, which is used for managing file permissions for individual users.
7. Specify a user ID for the user by selecting **Specify user ID manually**. If the option is not selected, the next available user ID above 500 is assigned to the new user.
8. Click the **OK** button to complete the process.

Look at the sample **Add New User** dialog box configuration:



To configure more advanced user properties, such as password expiration, modify the user's properties after adding the user.

3.2.3. Modifying User Properties

1. Select the user from the user list by clicking once on the user name.

2. Click Properties from the toolbar or choose File → Properties from the drop-down menu.

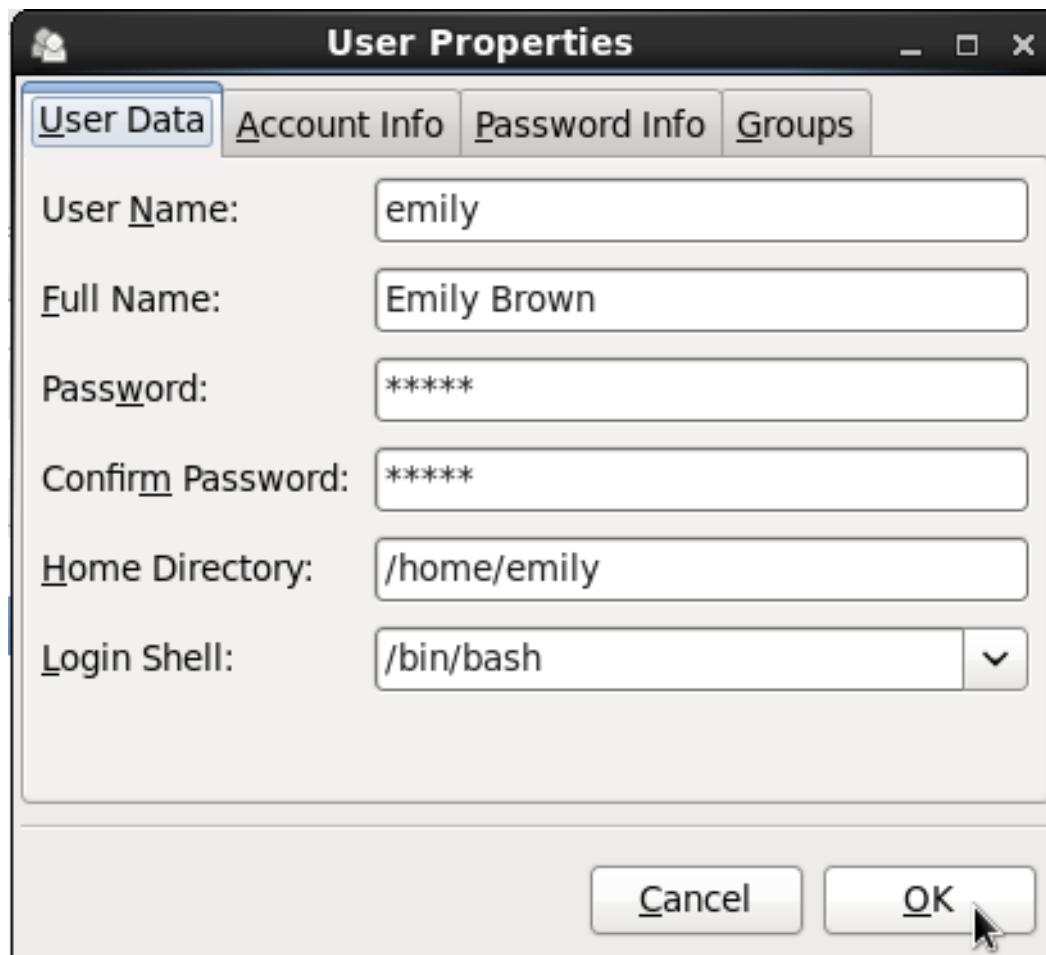


Figure 3.2. User Properties

3. There are four tabs you can update to your preferences. When you have finished, click the **OK** button to save your changes.

3.3. MANAGING GROUPS VIA THE USER MANAGER APPLICATION

3.3.1. Viewing Groups

In order to display the main window of **User Manager** to view groups, from the toolbar select **Edit → Preferences**. If you want to view *all* the groups, clear the **Hide system users and groups** check box.

The **Groups** tab provides a list of local groups with information about their group ID and group members as you can see in the picture below.

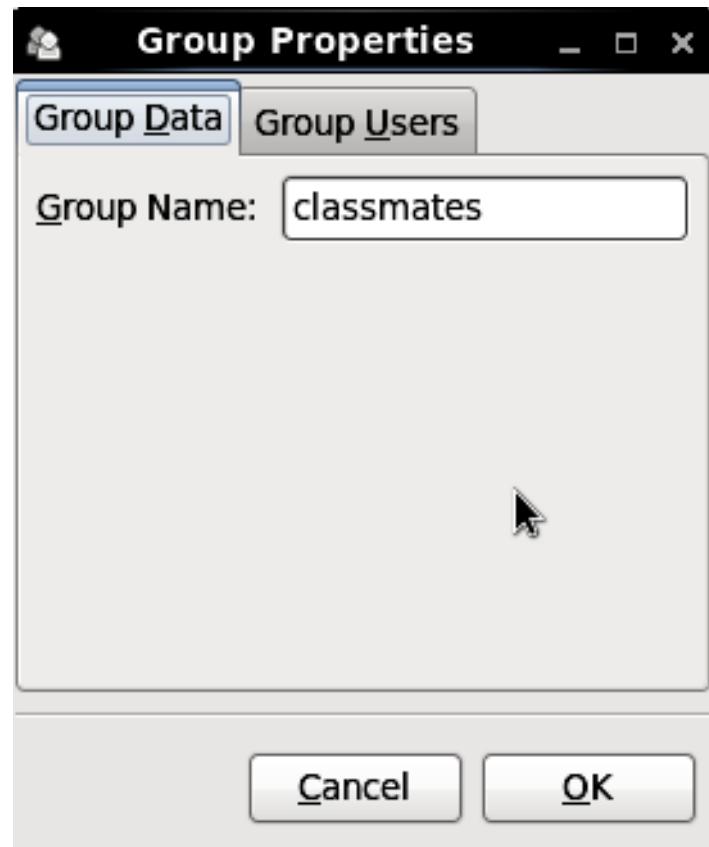


Figure 3.3. Viewing Groups

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

3.3.2. Adding a New Group

If there is a new group you need to add to the system, follow this procedure:

1. Select **Add Group** from the User Manager toolbar:

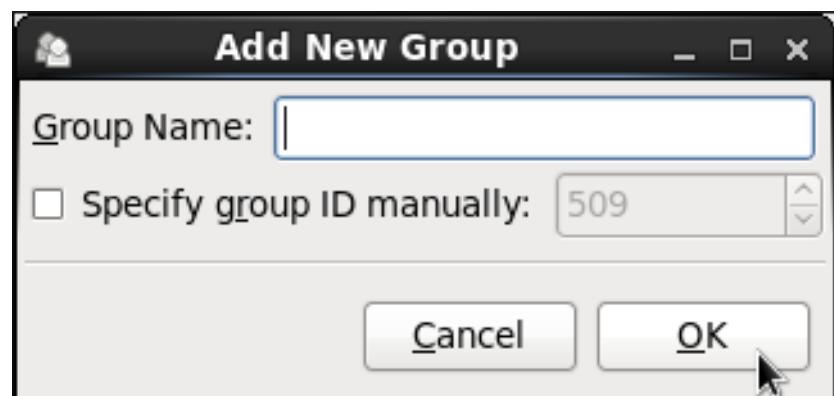


Figure 3.4. New Group

2. Type the name of the new group.
3. Specify the group ID (GID) for the new group by checking the **Specify group ID manually** check box.

4. Select the GID. Note that Red Hat Enterprise Linux also reserves group IDs lower than 500 for system groups.
5. Click **OK** to create the group. The new group appears in the group list.

3.3.3. Modifying Group Properties

1. Select the group from the group list by clicking on its name.
2. Click **Properties** from the toolbar or choose **File → Properties** from the drop-down menu.

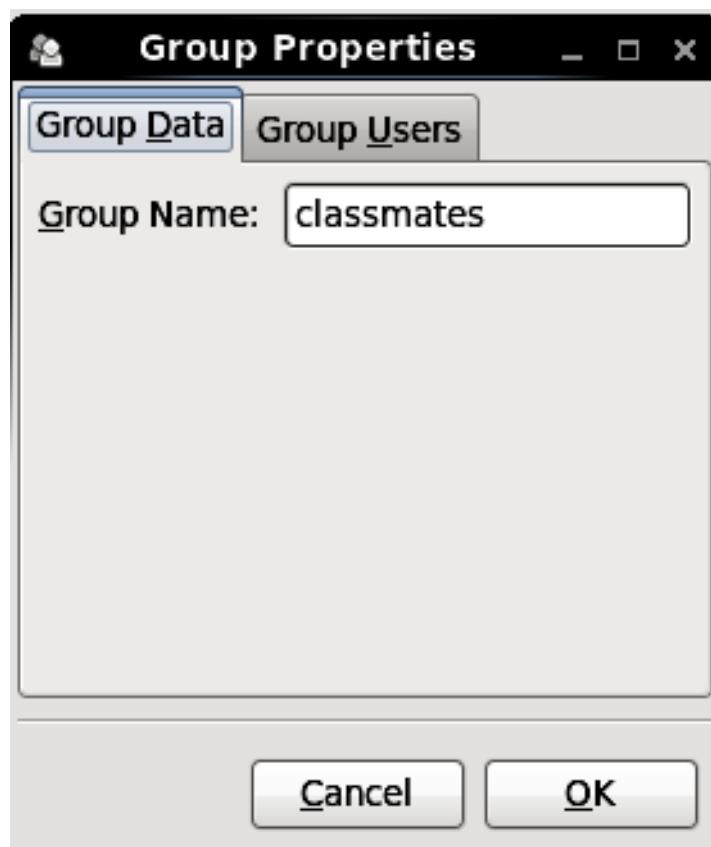


Figure 3.5. Group Properties

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

3.4. MANAGING USERS VIA COMMAND-LINE TOOLS

When managing users via command line, the following commands are used: **useradd**, **usermod**, **userdel**, or **passwd**. The files affected include **/etc/passwd** which stores user accounts information and **/etc/shadow**, which stores secure user account information.

3.4.1. Creating Users

The **useradd** utility creates new users and adds them to the system. Following the short procedure below, you will create a default user account with its UID, automatically create a home directory where default user settings will be stored, **/home/username/**, and set the default shell to **/bin/bash**.

1. Run the following command at a shell prompt as **root** substituting *username* with the name of your choice:

```
useradd username
```

2. By setting a password unlock the account to make it accessible. Type the password twice when the program prompts you to.

```
passwd
```

Example 3.1. Creating a User with Default Settings

```
~]# useradd robert
~]# passwd robert
Changing password for user robert
New password:
Re-type new password:
passwd: all authentication tokens updated successfully.
```

Running the **useradd robert** command creates an account named **robert**. If you run **cat /etc/passwd** to view the content of the **/etc/passwd** file, you can learn more about the new user from the line displayed to you:

```
robert:x:502:502::/home/robert:/bin/bash
```

robert has been assigned a UID of 502, which reflects the rule that the default UID values from 0 to 499 are typically reserved for system accounts. GID, group ID of **User Private Group**, equals to UID. The home directory is set to **/home/robert** and login shell to **/bin/bash**. The letter **x** signals that shadow passwords are used and that the hashed password is stored in **/etc/shadow**.

If you want to change the basic default setup for the user while creating the account, you can choose from a list of command-line options modifying the behavior of **useradd** (see the **useradd(8)** man page for the whole list of options). As you can see from the basic syntax of the command, you can add one or more options:

```
useradd [option(s)] username
```

As a system administrator, you can use the **-c** option to specify, for example, the full name of the user when creating them. Use **-c** followed by a string, which adds a comment to the user:

```
useradd -c "string" username
```

Example 3.2. Specifying a User's Full Name when Creating a User

```
~]# useradd -c "Robert Smith" robert
~]# cat /etc/passwd
robert:x:502:502:Robert Smith:/home/robert:/bin/bash
```

A user account has been created with user name **robert**, sometimes called the login name, and full name Robert Smith.

If you do not want to create the default **/home/username** directory for the user account, set a different one instead of it. Execute the command below:

```
useradd -d home_directory
```

Example 3.3. Adding a User with non-default Home Directory

```
~]# useradd -d /home/dir_1 robert
```

robert's home directory is now not the default **/home/robert** but **/home/dir_1/**.

If you do not want to create the home directory for the user at all, you can do so by running **useradd** with the **-M** option. However, when such a user logs into a system that has just booted and their home directory does not exist, their login directory will be the root directory. If such a user logs into a system using the **su** command, their login directory will be the current directory of the previous user.

```
useradd -M username
```

If you need to copy a directory content to the **/home** directory while creating a new user, make use of the **-m** and **-k** options together followed by the path.

Example 3.4. Creating a User while Copying Contents to the Home Directory

The following command copies the contents of a directory named **/dir_1** to **/home/jane**, which is the default home directory of a new user **jane**:

```
~]# useradd -m -k /dir_1 jane
```

As a system administrator, you may need to create a temporary account. Using the **useradd** command, this means creating an account for a certain amount of time only and disabling it at a certain date. This is a particularly useful setting as there is no security risk resulting from forgetting to delete a certain account. For this, the **-e** option is used with the specified `expire_date` in the YYYY-MM-DD format.



NOTE

Do not confuse account expiration and password expiration. Account expiration is a particular date, after which it is impossible to log in to the account in any way, as the account no longer exists. Password expiration, the maximum password age and date of password creation or last password change, is the date, when it is not possible to log in using the password (but other ways exist, such as logging in using an SSH key).

```
useradd -e YYYY-MM-DD username
```

Example 3.5. Setting the Account Expiration Date

```
~]# useradd -e 2015-11-05 emily
```

The account **emily** will be created now and automatically disabled on 5 November, 2015.

User's login shell defaults to **/bin/bash**, but can be changed by the **-s** option to any other shell different from bash, ksh, csh, tsh, for example.

```
useradd -s /login_shell username
```

Example 3.6. Adding a User with Non-default Shell

```
~]# useradd -s /bin/ksh robert
```

This command creates the user **robert** which has the **/bin/ksh** shell.

The **-r** option creates a system account, which is an account for administrative use that has some, but not all, root privileges. Such accounts have a UID lower than the value of *UID_MIN* defined in **/etc/login.defs**, typically 500 and above for ordinary users.

```
useradd -r username
```

3.4.2. Attaching New Users to Groups

The **useradd** command creates a **User Private Group** (*UPG*, a group assigned to a user account to which that user exclusively belongs) whenever a new user is added to the system and names the group after the user. For example, when the account **robert** is created, an UPG named **robert** is created at the same time, the only member of which is the user **robert**.

If you do not want to create a **User Private Group** for a user for whatever reason, execute the **useradd** command with the following option:

```
useradd -N username
```

Instead of automatically creating UPG or not creating it at all, you can specify the user's group membership with **-g** and **-G** options. While the **-g** option specifies the primary group membership, **-G** refers to supplementary groups into which the user is also included. The group names you specify must already exist on the system.

Example 3.7. Adding a User to a Group

```
~]# useradd -g "friends" -G "family,schoolmates" emily
```

The **useradd -g "friends" -G "family,schoolmates" emily** command creates the user **emily** but **emily**'s primary group is set to **friends** as specified by the **-g** option. **emily** is also a group member of the supplementary groups **family** and **schoolmates**.

Provided the user already exists and you want to add them to certain supplementary group(s), use the **usermod** command with the **-G** option and a list of groups divided by commas, no spaces:

```
usermod -G group_1,group_2,group_3
```

3.4.3. Updating Users' Authentication

When running the basic **useradd username** command, the password is automatically set to never expire (see the **/etc/shadow** file).

If you want to change this, use **passwd**, the standard utility for administering the **/etc/passwd** file. The syntax of the **passwd** command look as follows:

```
passwd option(s) username
```

You can, for example, lock the specified account. The locking is performed by rendering the encrypted password into an invalid string by prefixing the encrypted string with an exclamation mark (!). If you later find a reason to unlock the account, **passwd** has a reverse operation for locking. Only **root** can carry out these two operations.

```
passwd -l username  
passwd -u username
```

Example 3.8. Unlocking a User Password

```
~]# passwd -l robert  
Locking password for user robert.  
passwd: Success  
~]# passwd -u robert  
passwd: Warning: unlocked password would be empty  
passwd: Unsafe operation (use -f to force)
```

At first, the **-l** option locks **robert**'s account password successfully. However, running the **passwd -u** command does not unlock the password because by default **passwd** refuses to create a passwordless account.

If you want a password for an account to expire, run **passwd** with the **-e** option. The user will be forced to change the password during the next login attempt:

```
passwd -e username
```

As far as the password lifetime is concerned, setting the minimum time between password changes is useful for forcing the user to really change the password. The system administrator can set the minimum (the **-n** option) and the maximum (the **-x** option) lifetimes. To inform the user about their password expiration, use the **-w** option. All these options must be accompanied with the number of days and can be run as **root** only.

Example 3.9. Adjusting Aging Data for User Passwords

```
~]# passwd -n 10 -x 60 -w 3 jane
```

The above command has set the minimum password lifetime to 10 days, the maximum password lifetime to 60, and the number of days **jane** will begin receiving warnings in advance that her password will expire to 3 day.

Later, when you cannot remember the password setting, make use of the **-S** option which outputs a short information for you to know the status of the password for a given account:

```
~]# passwd -S jane  
jane LK 2014-07-22 10 60 3 -1 (Password locked.)
```

You can also set the number of days after a password expires with the **useradd** command, which disables the account permanently. A value of **0** disables the account as soon as the password has expired, and a value of **-1** disables the feature, that is, the user will have to change his password when the password expires. The **-f** option is used to specify the number of days after a password expires until the account is disabled (but may be unblocked by system administrator):

```
useradd -f number-of-days username
```

For more information on the **passwd** command see the **passwd(1)** man page.

3.4.4. Modifying User Settings

When a user already exists and you need to specify any of the options now, use the **usermod** command. The logic of using **usermod** is identical to **useradd** as well as its syntax:

```
usermod option(s) username
```

If you need to change the user's user name, use the **-l** option with the new user name (or login).

Example 3.10. Changing User's Login

```
~]# usermod -l "emily-smith" emily
```

The **-l** option changes the name of the user from the login **emily** to the new login, **emily-smith**. Nothing else is changed. In particular, **emily**'s home directory name (**/home/emily**) remains the same unless it is changed manually to reflect the new user name.

In a similar way you can change the user's UID or user's home directory. See the example below:



NOTE

Find all files owned by the specified UID in system and change their owner. Do the same for Access Control List (ACL) referring to the UID. It is recommended to check there are no running processes as they keep running with the old UID.

Example 3.11. Changing User's UID and Home Directory

```
~]# usermod -a -u 699 -d /home/dir_2 robert
```

The command with **-a -u** and **-d** options changes the settings of user **robert**. Now, his ID is 699 instead of 501, and his home directory is no longer **/home/robert** but **/home/dir_2**.

With the **usermod** command you can also move the content of the user's home directory to a new location, or lock the account by locking its password.

Example 3.12. Changing User's

```
~]# usermod -m -d /home/jane -L jane
```

In this sample command, the **-m** and **-d** options used together move the content of **jane**'s home directory to the **/home/dir_3** directory. The **-L** option locks the access to **jane**'s account by locking its password.

For the whole list of options to be used with the **usermod** command, see the **usermod(8)** man page or run **usermod --help** on the command line.

3.4.5. Deleting Users

If you want to remove a user account from the system, use the **userdel** command on the command line as **root**.

userdel username

Combining **userdel** with the **-r** option removes files in the user's home directory along with the home directory itself and the user's mail spool. Files located in other file systems have to be searched for and deleted manually.

userdel -r username



NOTE

The **-r** option is relatively safer, and thus recommended, compared to **-f** which forces the removal of the user account even if the user is still logged in.

3.4.6. Displaying Comprehensive User Information

When administering users and groups on your system, you need a good tool to monitor their configuration and activity on the system. Red Hat Enterprise Linux 6 provides you with the **lslogins** command-line utility, which gives you a comprehensive overview of users and groups, not only regarding user or group account configuration but also their activity on the system.

The general syntax of **lslogins** is the following:

lslogins [OPTIONS]

where **OPTIONS** can be one or more available options and their related parameters. See the **lslogins(1)** manual page or the output of the **lslogins --help** command for the complete list of available options and their usage.

The **lslogins** utility gives versatile information in a variety of formats based on the chosen options. The following examples introduce the most basic as well as some of the most useful combinations.

Running the **lslogins** command without any options shows default information about all system and user accounts on the system. Specifically, their UID, user name, and GECOS information, as well as information about the user's last login to the system, and whether their password is locked or login by password disabled.

Example 3.13. Displaying basic information about all accounts on the system

```
~]# lslogins
UID USER      PWD-LOCK PWD-DENY LAST-LOGIN GECOS
```

0 root	0	0	root
1 bin	0	1	bin
2 daemon	0	1	daemon
3 adm	0	1	adm
4 lp	0	1	lp
5 sync	0	1	sync
6 shutdown	0	1	Jul21/16:20 shutdown
7 halt	0	1	halt
8 mail	0	1	mail
10 uucp	0	1	uucp
11 operator	0	1	operator
12 games	0	1	games
13 gopher	0	1	gopher
14 ftp	0	1	FTP User
29 rpcuser	0	1	RPC Service User
32 rpc	0	1	Rpcbind Daemon
38 ntp	0	1	
42 gdm	0	1	
48 apache	0	1	Apache
68 haldaemon	0	1	HAL daemon
69 vcsa	0	1	virtual console memory owner
72 tcpdump	0	1	
74 sshd	0	1	Privilege-separated SSH
81 dbus	0	1	System message bus
89 postfix	0	1	
99 nobody	0	1	Nobody
113 usbmuxd	0	1	usbmuxd user
170 avahi-autoipd	0	1	Avahi IPv4LL Stack
173 abrt	0	1	
497 pulse	0	1	PulseAudio System Daemon
498 saslauth	0	1	Saslauthd user
499 rtkit	0	1	RealtimeKit
500 jsmith	0	0	10:56:12 John Smith
501 jdoe	0	0	12:13:53 John Doe
502 esmith	0	0	12:59:05 Emily Smith
503 jeyre	0	0	12:22:14 Jane Eyre
65534 nfsnobody	0	1	Anonymous NFS User

To display detailed information about a single user, run the **lslogins LOGIN** command, where *LOGIN* is either a UID or a user name. The following example displays detailed information about **John Doe**'s account and his activity on the system:

Example 3.14. Displaying detailed information about a single account

```
~]# lslogins jdoe
Username:          jdoe
UID:              501
Gecos field:      John Doe
Home directory:   /home/jdoe
Shell:            /bin/bash
No login:         no
Password is locked: no
Password no required: no
Login by password disabled: no
```

```

Primary group:          jdoe
GID:                  501
Supplementary groups:   users
Supplementary group IDs: 100
Last login:            12:13:53
Last terminal:         pts/3
Last hostname:         192.168.100.1
Hushed:                no
Password expiration warn interval: 7
Password changed:      Aug01/02:00
Maximal change time:   99999
Password expiration:   Sep01/02:00
Selinux context:       unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

```

If you use the **--logins=LOGIN** option, you can display information about a group of accounts that are specified as a list of UIDs or user names. Specifying the **--output=COLUMNS** option, where *COLUMNS* is a list of available output parameters, you can customize the output of the **lslogins** command. For example, the following command shows login activity of the users root, jsmith, jdoe, and esmith:

Example 3.15. Displaying specific information about a group of users

```

~]# lslogins --logins=0,500,jdoe,esmith \
> --output=UID,USER,LAST-LGIN,LAST-TTY,FAILED-LGIN,FAILED-TTY
UID USER  LAST-LGIN LAST-TTY FAILED-LGIN FAILED-TTY
0 root
500 jsmith 10:56:12 pts/2
501 jdoe 12:13:53 pts/3
502 esmith 15:46:16 pts/3 15:46:09 ssh:notty

```

The **lslogins** utility also distinguishes between system and user accounts. To address system accounts in your query, use the **--system-accts** option. To address user accounts, use the **--user-accts**. For example, the following command displays information about supplementary groups and password expirations for all user accounts:

Example 3.16. Displaying information about supplementary groups and password expiration for all user accounts

```

~]# lslogins --user-accts --supp-groups --acc-expiration
UID USER    GID GROUP    SUPP-GIDS SUPP-GROUPS PWD-WARN PWD-MIN PWD-
MAX PWD-CHANGE
PWD-EXPIR
0 root     0 root        7      99999 Jul21/02:00
500 jsmith 500 jsmith  1000,100 staff,users 7      99999 Jul21/02:00
501 jdoe   501 jdoe    100    users       7      99999 Aug01/02:00
Sep01/02:00
502 esmith 502 esmith  100    users       7      99999 Aug01/02:00
503 jeyre  503 jeyre  1000,100 staff,users 7      99999 Jul28/02:00
Sep01/02:00
65534 nfsnobody 65534 nfsnobody           Jul21/02:00

```

The ability to format the output of **lslogins** commands according to the user's needs makes **lslogins** an

ideal tool to use in scripts and for automatic processing. For example, the following command returns a single string that represents the time and date of the last login. This string can be passed as input to another utility for further processing.

Example 3.17. Displaying a single piece of information without the heading

```
~]# lslogins --logins=jsmith --output=LAST-LOGIN --time-format=iso | tail -1  
2014-08-06T10:56:12+0200
```

3.5. MANAGING GROUPS VIA COMMAND-LINE TOOLS

Groups are a useful tool for permitting co-operation between different users. There is a set of commands for operating with groups such as **groupadd**, **groupmod**, **groupdel**, or **gpasswd**. The files affected include **/etc/group** which stores group account information and **/etc/gshadow**, which stores secure group account information.

3.5.1. Creating Groups

To add a new group to the system with default settings, the **groupadd** command is run at the shell prompt as **root**.

```
groupadd group_name
```

Example 3.18. Creating a Group with Default Settings

```
~]# groupadd friends
```

The **groupadd** command creates a new group called **friends**. You can read more information about the group from the newly-created line in the **/etc/group** file:

```
classmates:x:30005:
```

Automatically, the group **friends** is attached with a unique GID (group ID) of 30005 and is not attached with any users. Optionally, you can set a password for a group by running **gpasswd groupname**.

Alternatively, you can add command options with specific settings.

```
groupadd option(s) groupname
```

If you, for example, want to specify the numerical value of the group's ID (GID) when creating the group, run the **groupadd** command with the **-g** option. Remember that this value must be unique (unless the **-o** option is used) and the value must be non-negative.

```
groupadd -g GID
```

Example 3.19. Creating a Group with Specified GID

The command below creates a group named **schoolmates** and sets GID of 60002 for it:

```
~]# groupadd -g 60002 schoolmates
```

When used with **-g** and GID already exists, **groupadd** refuses to create another group with existing GID. As a workaround, use the **-f** option, with which **groupadd** creates a group, but with a different GID.

```
groupadd -f GID
```

You may also create a system group by attaching the **-r** option to the **groupadd** command. System groups are used for system purposes, which practically means that GID is allocated from 1 to 499 within the reserved range of 999.

```
groupadd -r group_name
```

For more information on **groupadd**, see the **groupadd(8)** man pages.

3.5.2. Attaching Users to Groups

If you want to add an existing user to the named group, you can make use of the **gpasswd** command.

```
gpasswd -a username which_group_to_edit
```

To remove a user from the named group, run:

```
gpasswd -d username which_group_to_edit
```

To set the list of group members, write the user names after the **--members** option dividing them with commas and no spaces:

```
gpasswd --members username_1,username_2 which_group_to_edit
```

3.5.3. Updating Group Authentication

The **gpasswd** command administers **/etc/group** and **/etc/gshadow** files. Note that this command works only if run by a group administrator.

Who is a group administrator? A group administrator can add and delete users as well as set, change, or remove the group password. A group can have more than one group administrator. The **root** user can add group administrators with the **gpasswd -A users groupname** where **users** is a comma-separated list of existing users you want to be group administrators (without any spaces between commas).

For changing a group's password, run the **gpasswd** command with the relevant group name. You will be prompted to type the new password of the group.

```
gpasswd groupname
```

Example 3.20. Changing a Group Password

```
~]# gpasswd crowd
Changing password for group crowd
New password:
```

Re-enter new password:

The password for the group **crowd** has been changed.

You can also remove the password from the named group by using the **-r** option.

`gpasswd -r schoolmates`

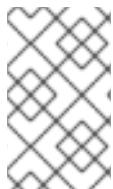
3.5.4. Modifying Group Settings

When a group already exists and you need to specify any of the options now, use the **groupmod** command. The logic of using **groupmod** is identical to **groupadd** as well as its syntax:

`groupmod option(s) groupname`

To change the group ID of a given group, use the **groupmod** command in the following way:

`groupmod -g GID_NEW which_group_to_edit`



NOTE

Find all files owned by the specified GID in system and change their owner. Do the same for Access Control List (ACL) referring to the GID. It is recommended to check there are no running processes as they keep running with the old GID.

To change the name of the group, run the following on the command line. The name of the group will be changed from *GROUP_NAME* to *NEW_GROUP_NAME* name.

`groupmod -n new_groupname groupname`

Example 3.21. Changing a Group's Name

The following command changes the name of the group **schoolmates** to **crowd**:

`~]# groupmod -n crowd schoolmates`

3.5.5. Deleting Groups

The **groupdel** command modifies the system account files, deleting all entries that see the group. The named group must exist when you execute this command.

`groupdel groupname`

3.6. ADDITIONAL RESOURCES

See the following resources for more information about managing users and groups.

3.6.1. Installed Documentation

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

- **chage(1)** – A command to modify password aging policies and account expiration.
- **gpasswd(1)** – A command to administer the **/etc/group** file.
- **groupadd(8)** – A command to add groups.
- **grpck(8)** – A command to verify the **/etc/group** file.
- **groupdel(8)** – A command to remove groups.
- **groupmod(8)** – A command to modify group membership.
- **pwck(8)** – A command to verify the **/etc/passwd** and **/etc/shadow** files.
- **pwconv(8)** – A tool to convert standard passwords to shadow passwords.
- **pwunconv(8)** – A tool to convert shadow passwords to standard passwords.
- **useradd(8)** – A command to add users.
- **userdel(8)** – A command to remove users.
- **usermod(8)** – A command to modify users.

For information about related configuration files, see:

- **group(5)** – The file containing group information for the system.
- **passwd(5)** – The file containing user information for the system.
- **shadow(5)** – The file containing passwords and account expiration information for the system.
- **login.defs(5)** – The file containing shadow password suite configuration.
- **useradd(8)** – For **/etc/default/useradd**, section “Changing the default values” in manual page.

CHAPTER 4. GAINING PRIVILEGES

System administrators (and in some cases users) will need to perform certain tasks with administrative access. Accessing the system as **root** is potentially dangerous and can lead to widespread damage to the system and data. This chapter covers ways to gain administrative privileges using the **su** and **sudo** programs. 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 6 Security Guide* for more information on administrative controls, potential dangers and ways to prevent data loss resulting from improper use of privileged access.

4.1. THE su COMMAND

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

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

Because this program is so powerful, administrators within an organization may 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 **User Manager** to modify group memberships, as follows. Note: you need Administrator privileges to perform this procedure.

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

See [Section 3.2, "Managing Users via the User Manager Application"](#) for more information about the **User Manager**.

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

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

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



NOTE

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

4.2. THE SUDO COMMAND

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

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

```
sudo <command>
```

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

The **sudo** command allows for a high degree of flexibility. For instance, only users listed in the **/etc/sudoers** configuration file are allowed to use the **sudo** command and the command is executed in *the user's shell*, not a root shell. This means the root shell can be completely disabled as shown in the *Red Hat Enterprise Linux 6 Security Guide*.

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

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

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

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

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

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

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

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

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

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

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

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

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



IMPORTANT

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

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

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

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

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

```
| sudo /bin/bash
```

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

4.3. ADDITIONAL RESOURCES

While programs allowing users to gain administrative privileges are a potential security risk, security itself is beyond the scope of this particular book. You should therefore see sources 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 as well as a list of options available for customizing **sudo**'s behavior.
- **pam(8)** – the manual page describing the use of Pluggable Authentication Modules for Linux.

Online Documentation

- [Red Hat Enterprise Linux 6 Security Guide](#) – The *Security Guide* describes in detail security risks and mitigating techniques related to programs for gaining privileges.

- [Red Hat Enterprise Linux 6 Managing Single Sign-On and Smart Cards](#) - This guide provides, among other things, a detailed description of *Pluggable Authentication Modules (PAM)*, their configuration and usage.

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

CHAPTER 5. CONSOLE ACCESS

When normal (non-root) users log into a computer locally, they are given two types of special permissions:

1. They can run certain programs that they otherwise cannot run.
2. They can access certain files that they otherwise cannot access. These files normally include special device files used to access diskettes, CD-ROMs, and so on.

Since there are multiple consoles on a single computer and multiple users can be logged into the computer locally at the same time, one of the users has to essentially win the race to access the files. The first user to log in at the console owns those files. Once the first user logs out, the next user who logs in owns the files.

In contrast, every user who logs in at the console is allowed to run programs that accomplish tasks normally restricted to the root user. If X is running, these actions can be included as menu items in a graphical user interface. As shipped, these console-accessible programs include **halt**, **poweroff**, and **reboot**.

5.1. DISABLING CONSOLE PROGRAM ACCESS FOR NON-ROOT USERS

Non-root users can be denied console access to any program in the **/etc/security/console.apps/** directory. To list these programs, run the following command:

```
~]$ ls /etc/security/console.apps
abrt-cli-root
config-util
eject
halt
poweroff
reboot
rhn_register
setup
subscription-manager
subscription-manager-gui
system-config-network
system-config-network-cmd
xserver
```

For each of these programs, console access denial can be configured using the program's *Pluggable Authentication Module* (PAM) configuration file. For information about PAMs and their usage, see chapter [Pluggable Authentication Modules](#) of the Red Hat Enterprise Linux 6 *Managing Single Sign-On and Smart Cards* guide.

PAM configuration file for each program in **/etc/security/console.apps/** resides in the **/etc/pam.d/** directory and is named the same as the program. Using this file, you can configure PAM to deny access to the program if the user is not root. To do that, insert line **auth requisite pam_deny.so** directly after the first uncommented line **auth sufficient pam_rootok.so**.

Example 5.1. Disabling Access to the Reboot Program

To disable non-root console access to **/etc/security/console.apps/reboot**, insert line **auth requisite pam_deny.so** into the **/etc/pam.d/reboot** PAM configuration file:

```
 #%PAM-1.0
auth sufficient pam_rootok.so
auth requisite pam_deny.so
auth required pam_console.so
#auth include system-auth
account required pam_permit.so
```

With this setting, all non-root access to the **reboot** utility is disabled.

Additionally, several programs in **/etc/security/console.apps** partially derive their PAM configuration from the **/etc/pam.d/config-util** configuration file. This allows to change configuration for all these programs at once by editing **/etc/pam.d/config-util**. To find all these programs, search for PAM configuration files that refer to the **config-util** file:

```
~]# grep -l "config-util" /etc/pam.d/*
/etc/pam.d/abrt-cli-root
/etc/pam.d/rhn_register
/etc/pam.d/subscription-manager
/etc/pam.d/subscription-manager-gui
/etc/pam.d/system-config-network
/etc/pam.d/system-config-network-cmd
```

Disabling console program access as described above may be useful in environments where the console is otherwise secured. Security measures may include password protection for BIOS and boot loader, disabling rebooting on pressing Ctrl+Alt+Delete, disabling the power and reset switches, and other. In these cases, you may want to restrict normal user's access to **halt**, **poweroff**, **reboot**, and other programs, which by default are accessible from the console.

5.2. DISABLING REBOOTING USING CTRL+ALT+DEL

The action that happens in response to pressing Ctrl+Alt+Del at console is specified in the **/etc/init/control-alt-delete.conf** file. By default, the **shutdown** utility with the **-r** option is used to shutdown and reboot the system.

To disable this action, create an overriding configuration file that specifies the **exec true** command, which does nothing. To do that, run the following command as root:

```
~]# echo "exec true" >> /etc/init/control-alt-delete.override
```

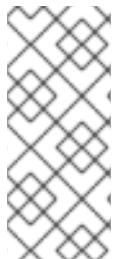
PART II. SUBSCRIPTION AND SUPPORT

To receive updates to the software on a Red Hat Enterprise Linux system it must be subscribed to the *Red Hat Content Delivery Network* (CDN) and the appropriate repositories enabled. This part describes how to subscribe a system to the Red Hat Content Delivery Network.

Red Hat provides support via the [Customer Portal](#), and you can access this support directly from the command line using the **Red Hat Support Tool**. This part describes the use of this command-line tool.

CHAPTER 6. REGISTERING THE SYSTEM AND MANAGING SUBSCRIPTIONS

The subscription service provides a mechanism to handle Red Hat software inventory and allows you to install additional software or update already installed programs to newer versions using the **yum** or **PackageKit** package managers. In Red Hat Enterprise Linux 6 the recommended way to register your system and attach subscriptions is to use *Red Hat Subscription Management*.



NOTE

It is also possible to register the system and attach subscriptions after installation during the firstboot process. For detailed information about firstboot see the [Firstboot](#) chapter in the [Installation Guide](#) for Red Hat Enterprise Linux 6. Note that firstboot is only available on systems after a graphical installation or after a kickstart installation where a desktop and the X window system were installed and graphical login was enabled.

6.1. REGISTERING THE SYSTEM AND ATTACHING SUBSCRIPTIONS

Complete the following steps to register your system and attach one or more subscriptions using Red Hat Subscription Management. Note that all **subscription-manager** commands are supposed to be run as **root**.

1. Run the following command to register your system. You will be prompted to enter your user name and password. Note that the user name and password are the same as your login credentials for Red Hat Customer Portal.

```
subscription-manager register
```

2. Determine the pool ID of a subscription that you require. To do so, type the following at a shell prompt to display a list of all subscriptions that are available for your system:

```
subscription-manager list --available
```

For each available subscription, this command displays its name, unique identifier, expiration date, and other details related to your subscription. To list subscriptions for all architectures, add the **--all** option. The pool ID is listed on a line beginning with **Pool ID**.

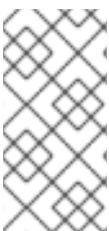
3. Attach the appropriate subscription to your system by entering a command as follows:

```
subscription-manager attach --pool=pool_id
```

Replace *pool_id* with the pool ID you determined in the previous step.

To verify the list of subscriptions your system has currently attached, at any time, run:

```
subscription-manager list --consumed
```



NOTE

If you use a firewall or a proxy, you may need additional configuration to allow **yum** and **subscription-manager** to work correctly. Refer to the "["Setting Firewall Access for Content Delivery"](#)" section of the Red Hat Enterprise Linux 6 Subscription Management guide if you use a firewall and to the "["Using an HTTP Proxy"](#)" section if you use a proxy.

For more details on how to register your system using Red Hat Subscription Management and associate it with subscriptions, see the designated [solution article](#). For comprehensive information about subscriptions, see the [Red Hat Subscription Management](#) collection of guides.

6.2. MANAGING SOFTWARE REPOSITORIES

When a system is subscribed to the Red Hat Content Delivery Network, a repository file is created in the **/etc/yum.repos.d/** directory. To verify that, use **yum** to list all enabled repositories:

```
yum repolist
```

Red Hat Subscription Management also allows you to manually enable or disable software repositories provided by Red Hat. To list all available repositories, use the following command:

```
subscription-manager repos --list
```

The repository names depend on the specific version of Red Hat Enterprise Linux you are using and are in the following format:

```
rhel-variant-rhscl-version-rpms  
rhel-variant-rhscl-version-debug-rpms  
rhel-variant-rhscl-version-source-rpms
```

Where *variant* is the Red Hat Enterprise Linux system variant (**server** or **workstation**), and *version* is the Red Hat Enterprise Linux system version (**6** or **7**), for example:

```
rhel-server-rhscl-6-eus-rpms  
rhel-server-rhscl-6-eus-source-rpms  
rhel-server-rhscl-6-eus-debug-rpms
```

To enable a repository, enter a command as follows:

```
subscription-manager repos --enable repository
```

Replace *repository* with a name of the repository to enable.

Similarly, to disable a repository, use the following command:

```
subscription-manager repos --disable repository
```

[Section 8.4, "Configuring Yum and Yum Repositories"](#) provides detailed information about managing software repositories using **yum**.

6.3. REMOVING SUBSCRIPTIONS

To remove a particular subscription, complete the following steps.

1. Determine the serial number of the subscription you want to remove by listing information about already attached subscriptions:

subscription-manager list --consumed

The serial number is the number listed as **serial**. For instance, **744993814251016831** in the example below:

SKU:	ES0113909
Contract:	01234567
Account:	1234567
Serial:	744993814251016831
Pool ID:	8a85f9894bba16dc014bccdd905a5e23
Active:	False
Quantity Used:	1
Service Level:	SELF-SUPPORT
Service Type:	L1-L3
Status Details:	
Subscription Type:	Standard
Starts:	02/27/2015
Ends:	02/27/2016
System Type:	Virtual

2. Enter a command as follows to remove the selected subscription:

subscription-manager remove --serial=serial_number

Replace *serial_number* with the serial number you determined in the previous step.

To remove all subscriptions attached to the system, run the following command:

subscription-manager remove --all

6.4. ADDITIONAL RESOURCES

For more information on how to register your system using Red Hat Subscription Management and associate it with subscriptions, see the resources listed below.

Installed Documentation

- **subscription-manager(8)** – the manual page for Red Hat Subscription Management provides a complete list of supported options and commands.

Related Books

- [Red Hat Subscription Management](#) collection of guides – These guides contain detailed information how to use Red Hat Subscription Management.
- [Installation Guide](#) – see the [Firstboot](#) chapter for detailed information on how to register during the firstboot process.

Online Resources

- [Red Hat Access Labs](#) – The Red Hat Access Labs includes a “Registration Assistant”.

See Also

- [Chapter 4, Gaining Privileges](#) documents how to gain administrative privileges by using the **su** and **sudo** commands.
- [Chapter 8, Yum](#) provides information about using the **yum** packages manager to install and update software.
- [Chapter 9, PackageKit](#) provides information about using the **PackageKit** package manager to install and update software.

CHAPTER 7. ACCESSING SUPPORT USING THE RED HAT SUPPORT TOOL

The **Red Hat Support Tool**, in the redhat-support-tool package, can function as both an interactive shell and as a single-execution program. It can be run over **SSH** or from any terminal. It enables, for example, searching the Red Hat Knowledgebase from the command line, copying solutions directly on the command line, opening and updating support cases, and sending files to Red Hat for analysis.

7.1. INSTALLING THE RED HAT SUPPORT TOOL

The **Red Hat Support Tool** is installed by default on Red Hat Enterprise Linux. If required, to ensure that it is, enter the following command as **root**:

```
~]# yum install redhat-support-tool
```

7.2. REGISTERING THE RED HAT SUPPORT TOOL USING THE COMMAND LINE

To register the Red Hat Support Tool to the customer portal using the command line, proceed as follows:

1. ~]# redhat-support-tool config user *username*

Where *username* is the user name of the Red Hat Customer Portal account.

2. ~]# redhat-support-tool config password
Please enter the password for *username*:

7.3. USING THE RED HAT SUPPORT TOOL IN INTERACTIVE SHELL MODE

To start the tool in interactive mode, enter the following command:

```
~]$ redhat-support-tool  
Welcome to the Red Hat Support Tool.  
Command (? for help):
```

The tool can be run as an unprivileged user, with a consequently reduced set of commands, or as **root**.

The commands can be listed by entering the **?** character. The program or menu selection can be exited by entering the **q** or **e** character. You will be prompted for your Red Hat Customer Portal user name and password when you first search the Knowledgebase or support cases. Alternately, set the user name and password for your Red Hat Customer Portal account using interactive mode, and optionally save it to the configuration file.

7.4. CONFIGURING THE RED HAT SUPPORT TOOL

When in interactive mode, the configuration options can be listed by entering the command **config --help**:

```
~]# redhat-support-tool
Welcome to the Red Hat Support Tool.
Command (? for help): config --help

Usage: config [options] config.option <new option value>
```

Use the 'config' command to set or get configuration file values.

Options:

- h, --help show this help message and exit
- g, --global Save configuration option in /etc/redhat-support-tool.conf.
- u, --unset Unset configuration option.

The configuration file options which can be set are:

user : The Red Hat Customer Portal user.
 password : The Red Hat Customer Portal password.
 debug : CRITICAL, ERROR, WARNING, INFO, or DEBUG
 url : The support services URL. Default=https://api.access.redhat.com
 proxy_url : A proxy server URL.
 proxy_user: A proxy server user.
 proxy_password: A password for the proxy server user.
 ssl_ca : Path to certificate authorities to trust during communication.
 kern_debug_dir: Path to the directory where kernel debug symbols should be downloaded and cached. Default=/var/lib/redhat-support-tool/debugkernels

Examples:

- config user
- config user my-rhn-username
- config --unset user

Procedure 7.1. Registering the Red Hat Support Tool Using Interactive Mode

To register the Red Hat Support Tool to the customer portal using interactive mode, proceed as follows:

1. Start the tool by entering the following command:

```
~]# redhat-support-tool
```

2. Enter your Red Hat Customer Portal user name:

```
Command (? for help): config user username
```

To save your user name to the global configuration file, add the **-g** option.

3. Enter your Red Hat Customer Portal password:

```
Command (? for help): config password
```

Please enter the password for *username*:

7.4.1. Saving Settings to the Configuration Files

The **Red Hat Support Tool** unless otherwise directed, stores values and options locally in the home directory of the current user, using the **~/.redhat-support-tool/redhat-support-tool.conf** configuration file. If required, it is recommended to save passwords to this file because it is only readable by that

particular user. When the tool starts, it will read values from the global configuration file **/etc/redhat-support-tool.conf** and from the local configuration file. Locally stored values and options take precedence over globally stored settings.



WARNING

It is recommended **not** to save passwords in the global **/etc/redhat-support-tool.conf** configuration file because the password is just **base64** encoded and can easily be decoded. In addition, the file is world readable.

To save a value or option to the global configuration file, add the **-g, --global** option as follows:

Command (? for help): config *setting -g value*



NOTE

In order to be able to save settings globally, using the **-g, --global** option, the **Red Hat Support Tool** must be run as **root** because normal users do not have the permissions required to write to **/etc/redhat-support-tool.conf**.

To remove a value or option from the local configuration file, add the **-u, --unset** option as follows:

Command (? for help): config *setting -u value*

This will clear, unset, the parameter from the tool and fall back to the equivalent setting in the global configuration file, if available.



NOTE

When running as an unprivileged user, values stored in the global configuration file cannot be removed using the **-u, --unset** option, but they can be cleared, unset, from the current running instance of the tool by using the **-g, --global** option simultaneously with the **-u, --unset** option. If running as **root**, values and options can be removed from the global configuration file using **-g, --global** simultaneously with the **-u, --unset** option.

7.5. OPENING AND UPDATING SUPPORT CASES USING INTERACTIVE MODE

Procedure 7.2. Opening a New Support Case Using Interactive Mode

To open a new support case using interactive mode, proceed as follows:

1. Start the tool by entering the following command:

~]# redhat-support-tool

2. Enter the **opencase** command:

Command (? for help): opencase

3. Follow the on screen prompts to select a product and then a version.
4. Enter a summary of the case.
5. Enter a description of the case and press **Ctrl+D** on an empty line when complete.
6. Select a severity of the case.
7. Optionally chose to see if there is a solution to this problem before opening a support case.
8. Confirm you would still like to open the support case.

Support case 0123456789 has successfully been opened

9. Optionally chose to attach an SOS report.
10. Optionally chose to attach a file.

Procedure 7.3. Viewing and Updating an Existing Support Case Using Interactive Mode

To view and update an existing support case using interactive mode, proceed as follows:

1. Start the tool by entering the following command:

~]# redhat-support-tool

2. Enter the **getcase** command:

Command (? for help): getcase *case-number*

Where *case-number* is the number of the case you want to view and update.

3. Follow the on screen prompts to view the case, modify or add comments, and get or add attachments.

Procedure 7.4. Modifying an Existing Support Case Using Interactive Mode

To modify the attributes of an existing support case using interactive mode, proceed as follows:

1. Start the tool by entering the following command:

~]# redhat-support-tool

2. Enter the **modifycase** command:

Command (? for help): modifycase *case-number*

Where *case-number* is the number of the case you want to view and update.

3. The modify selection list appears:

Type the number of the attribute to modify or 'e' to return to the previous menu.

- 1 Modify Type
 - 2 Modify Severity
 - 3 Modify Status
 - 4 Modify Alternative-ID
 - 5 Modify Product
 - 6 Modify Version
- End of options.

Follow the on screen prompts to modify one or more of the options.

4. For example, to modify the status, enter **3**:

```
Selection: 3
1 Waiting on Customer
2 Waiting on Red Hat
3 Closed
Please select a status (or 'q' to exit):
```

7.6. VIEWING SUPPORT CASES ON THE COMMAND LINE

Viewing the contents of a case on the command line provides a quick and easy way to apply solutions from the command line.

To view an existing support case on the command line, enter a command as follows:

```
~]# redhat-support-tool getcase case-number
```

Where *case-number* is the number of the case you want to download.

7.7. ADDITIONAL RESOURCES

The Red Hat Knowledgebase article [Red Hat Support Tool](#) has additional information, examples, and video tutorials.

PART III. INSTALLING AND MANAGING SOFTWARE

All software on a Red Hat Enterprise Linux system is divided into RPM packages, which can be installed, upgraded, or removed. This part focuses on product subscriptions and entitlements, and describes how to manage packages on Red Hat Enterprise Linux using both **Yum** and the **PackageKit** suite of graphical package management tools.

CHAPTER 8. YUM

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

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

The following sections assume your system was registered with Red Hat Subscription Management during installation as described in the [Red Hat Enterprise Linux 6 Installation Guide](#). If your system is not subscribed, see [Chapter 6, Registering the System and Managing Subscriptions](#).



IMPORTANT

Yum provides secure package management by enabling GPG (Gnu Privacy Guard; also known as GnuPG) signature verification on GPG-signed packages to be turned on for all package repositories (i.e. package sources), or for individual repositories. When signature verification is enabled, Yum will refuse to install any packages not GPG-signed with the correct key for that repository. This means that you can trust that the **RPM** packages you download and install on your system are from a trusted source, such as Red Hat, and were not modified during transfer. See [Section 8.4, “Configuring Yum and Yum Repositories”](#) for details on enabling signature-checking with Yum, or [Section B.3, “Checking a Package’s Signature”](#) for information on working with and verifying GPG-signed **RPM** packages in general.

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

Learning Yum is a worthwhile investment because it is often the fastest way to perform system administration tasks, and it provides capabilities beyond those provided by the **PackageKit** graphical package management tools. See [Chapter 9, PackageKit](#) for details on using **PackageKit**.



NOTE

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

8.1. CHECKING FOR AND UPDATING PACKAGES

8.1.1. Checking For Updates

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

```
yum check-update
```

For example:

```

~]# yum check-update
Loaded plugins: product-id, refresh-packagekit, subscription-manager
Updating Red Hat repositories.
INFO:rhsm-app.repolib:repos updated: 0
PackageKit.x86_64          0.5.8-2.el6      rhel
PackageKit-glib.x86_64       0.5.8-2.el6      rhel
PackageKit-yum.x86_64        0.5.8-2.el6      rhel
PackageKit-yum-plugin.x86_64 0.5.8-2.el6      rhel
glibc.x86_64                2.11.90-20.el6    rhel
glibc-common.x86_64         2.10.90-22      rhel
kernel.x86_64               2.6.31-14.el6    rhel
kernel-firmware.noarch       2.6.31-14.el6    rhel
rpm.x86_64                 4.7.1-5.el6      rhel
rpm-libs.x86_64              4.7.1-5.el6      rhel
rpm-python.x86_64             4.7.1-5.el6      rhel
udev.x86_64                  147-2.15.el6     rhel
yum.noarch                   3.2.24-4.el6     rhel

```

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

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

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

8.1.2. Updating Packages

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

Updating a Single Package

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

```
yum update package_name
```

For example, to update the udev package, type:

```

~]# yum update udev
Loaded plugins: product-id, refresh-packagekit, subscription-manager
Updating Red Hat repositories.
INFO:rhsm-app.repolib:repos updated: 0
Setting up Update Process
Resolving Dependencies
--> Running transaction check
--> Package udev.x86_64 0:147-2.15.el6 set to be updated
--> Finished Dependency Resolution

```

Dependencies Resolved

```
=====
Package      Arch      Version       Repository   Size
=====
Updating:
udev        x86_64    147-2.15.el6    rhel        337 k
```

Transaction Summary

```
=====
Install      0 Package(s)
Upgrade      1 Package(s)
```

Total download size: 337 k

Is this ok [y/N]:

This output contains several items of interest:

- Loaded plugins: product-id, refresh-packagekit, subscription-manager** – **yum** always informs you which Yum plug-ins are installed and enabled. See [Section 8.5, “Yum Plug-ins”](#) for general information on Yum plug-ins, or to [Section 8.5.3, “Plug-in Descriptions”](#) for descriptions of specific plug-ins.
- udev.x86_64** – you can download and install new `udev` package.
- yum** presents the update information and then prompts you as to whether you want it to perform the update; **yum** runs interactively by default. If you already know which transactions the **yum** command plans to perform, you can use the `-y` option to automatically answer **yes** to any questions that **yum** asks (in which case it runs non-interactively). However, you should always examine which changes **yum** plans to make to the system so that you can easily troubleshoot any problems that might arise.

If a transaction does go awry, you can view Yum's transaction history by using the **yum history** command as described in [Section 8.3, “Working with Transaction History”](#).



IMPORTANT

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

When using **RPM**, on the other hand, it is important to use the **rpm -i kernel** command (which installs a new kernel) instead of **rpm -u kernel** (which replaces the current kernel). See [Section B.2.2, “Installing and Upgrading”](#) for more information on installing/upgrading kernels with **RPM**.

Updating All Packages and Their Dependencies

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

```
yum update
```

Updating Security-Related Packages

Discovering which packages have security updates available and then updating those packages quickly and easily is important. Yum provides the plug-in for this purpose. The **security** plug-in extends the **yum** command with a set of highly-useful security-centric commands, subcommands and options. See [Section 8.5.3, “Plug-in Descriptions”](#) for specific information.

Updating Packages Automatically

It is also possible to set up periodical automatic updates for your packages. For this purpose, Red Hat Enterprise Linux 6 uses the yum-cron package. It provides a Yum interface for the **cron** daemon and downloads metadata from your package repositories. With the **yum-cron** service enabled, the user can schedule an automated daily Yum update as a cron job.



NOTE

The yum-cron package is provided by the Optional subscription channel. See [Section 8.4.8, “Adding the Optional and Supplementary Repositories”](#) for more information on Red Hat additional channels.

To install **yum-cron** issue the following command:

```
~]# yum install yum-cron
```

By default, the **yum-cron** service is disabled and needs to be activated and started manually:

```
~]# chkconfig yum-cron on
```

```
~]# service yum-cron start
```

To verify the status of the service, run the following command:

```
~]# service yum-cron status
```

The script included in the yum-cron package can be configured to change the extent and frequency of the updates, as well as to send notifications to e-mail. To customize **yum-cron**, edit the **/etc/sysconfig/yum-cron** file.

Additional details and instructions for **yum-cron** can be found in the comments within **/etc/sysconfig/yum-cron** and at the **yum-cron(8)** manual page.

8.1.3. Preserving Configuration File Changes

You will inevitably make changes to the configuration files installed by packages as you use your Red Hat Enterprise Linux system. **RPM**, which Yum uses to perform changes to the system, provides a mechanism for ensuring their integrity. See [Section B.2.2, “Installing and Upgrading”](#) for details on how to manage changes to configuration files across package upgrades.

8.1.4. Upgrading the System Off-line with ISO and Yum

For systems that are disconnected from the Internet or Red Hat Network, using the **yum update** command with the Red Hat Enterprise Linux installation ISO image is an easy and quick way to upgrade systems to the latest minor version. The following steps illustrate the upgrading process:

1. Create a target directory to mount your ISO image. This directory is not automatically created when mounting, so create it before proceeding to the next step. As **root**, type:

mkdir mount_dir

Replace *mount_dir* with a path to the mount directory. Typically, users create it as a subdirectory in the **/media** directory.

2. Mount the Red Hat Enterprise Linux 6 installation ISO image to the previously created target directory. As **root**, type:

mount -o loop iso_name mount_dir

Replace *iso_name* with a path to your ISO image and *mount_dir* with a path to the target directory. Here, the **-o loop** option is required to mount the file as a block device.

3. Copy the **media.repo** file from the mount directory to the **/etc/yum.repos.d/** directory. Note that configuration files in this directory must have the *.repo* extension to function properly.

cp mount_dir/media.repo /etc/yum.repos.d/new.repo

This creates a configuration file for the yum repository. Replace *new.repo* with the filename, for example *rhel6.repo*.

4. Edit the new configuration file so that it points to the Red Hat Enterprise Linux installation ISO. Add the following line into the **/etc/yum.repos.d/new.repo** file:

baseurl=file:///mount_dir

Replace *mount_dir* with a path to the mount point.

5. Update all yum repositories including **/etc/yum.repos.d/new.repo** created in previous steps. As **root**, type:

yum update

This upgrades your system to the version provided by the mounted ISO image.

6. After successful upgrade, you can unmount the ISO image. As **root**, type:

umount mount_dir

where *mount_dir* is a path to your mount directory. Also, you can remove the mount directory created in the first step. As **root**, type:

rmdir mount_dir

7. If you will not use the previously created configuration file for another installation or update, you can remove it. As **root**, type:

rm /etc/yum.repos.d/new.repo**Example 8.1. Upgrading from Red Hat Enterprise Linux 6.3 to 6.4**

Imagine you need to upgrade your system without access to the Internet. To do so, you want to use an ISO image with the newer version of the system, called for instance **RHEL6.4-Server-**

20130130.0-x86_64-DVD1.iso. A target directory created for mounting is **/media/rhel6/**. As **root**, change into the directory with your ISO image and type:

```
~]# mount -o loop RHEL6.4-Server-20130130.0-x86_64-DVD1.iso /media/rhel6/
```

Then set up a yum repository for your image by copying the **media.repo** file from the mount directory:

```
~]# cp /media/rhel6/media.repo /etc/yum.repos.d/rhel6.repo
```

To make yum recognize the mount point as a repository, add the following line into the **/etc/yum.repos.d/rhel6.repo** copied in the previous step:

```
baseurl=file:///media/rhel6/
```

Now, updating the yum repository will upgrade your system to a version provided by **RHEL6.4-Server-20130130.0-x86_64-DVD1.iso**. As **root**, execute:

```
~]# yum update
```

When your system is successfully upgraded, you can unmount the image, remove the target directory and the configuration file:

```
~]# umount /media/rhel6/
```

```
~]# rmdir /media/rhel6/
```

```
~]# rm /etc/yum.repos.d/rhel6.repo
```

8.2. PACKAGES AND PACKAGE GROUPS

8.2.1. Searching Packages

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

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

Replace *term* with a package name you want to search.

Example 8.2. Searching for packages matching a specific string

To list all packages that match "vim", "gvim", or "emacs", type:

```
~]$ yum search vim gvim emacs
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager
=====
N/S matched: vim =====
vim-X11.x86_64 : The VIM version of the vi editor for the X Window System
vim-common.x86_64 : The common files needed by any version of the VIM editor
[output truncated]
```

```
===== N/S matched: emacs =====
emacs.x86_64 : GNU Emacs text editor
emacs-auctex.noarch : Enhanced TeX modes for Emacs
[output truncated]
```

Name and summary matches mostly, use "search all" for everything.
Warning: No matches found for: gvim

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

8.2.2. Listing Packages

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

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



NOTE

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

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

See [Example 8.3, “Listing all ABRT add-ons and plug-ins using glob expressions”](#) and [Example 8.5, “Listing available packages using a single glob expression with escaped wildcard characters”](#) for an example usage of both these methods.

yum list *glob_expression*

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

Example 8.3. Listing all ABRT add-ons and plug-ins using glob expressions

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

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

abrt-plugin-bugzilla.x86_64	1.0.7-5.el6	@rhel
abrt-plugin-logger.x86_64	1.0.7-5.el6	@rhel
abrt-plugin-sosreport.x86_64	1.0.7-5.el6	@rhel
abrt-plugin-ticketuploader.x86_64	1.0.7-5.el6	@rhel

yum list all

Lists all installed *and* available packages.

yum 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 8.4. 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:

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

yum list available

Lists all available packages in all enabled repositories.

Example 8.5. 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:

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

yum grouplist

Lists all package groups.

yum repolist

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

8.2.3. Displaying Package Information

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

yum info package_name

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

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

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

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

yumdb info package_name

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

```
~]# yumdb info yum
Loaded plugins: product-id, refresh-packagekit, subscription-manager
yum-3.2.27-4.el6.noarch
checksum_data = 23d337ed51a9757bbfbdc82c4eaca9808ff1009b51e9626d540f44fe95f771
checksum_type = sha256
from_repo = rhel
```

```
from_repo_revision = 1298613159
from_repo_timestamp = 1298614288
installed_by = 4294967295
reason = user
releasever = 6.1
```

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

Listing Files Contained in a Package

repoquery is a program for querying information from yum repositories similarly to rpm queries. You can query both package groups and individual packages. To list all files contained in a specific package, type:

```
repoquery --list package_name
```

Replace *package_name* with a name of the package you want to inspect. For more information on the **repoquery** command, see the **repoquery** manual page.

To find out which package provides a specific file, you can use the **yum provides** command, described in [Finding which package owns a file](#)

8.2.4. Installing Packages

Yum 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:

```
yum install package_name
```

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

```
yum install package_name package_name
```

If you are installing packages on a *multilib* system, such as an AMD64 or Intel 64 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 `sqlite` package for **i686**, type:

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

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

```
~]# yum install perl-Crypt-*
```

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

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

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



NOTE

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

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

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

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*. The **yum grouplist -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** in the following example:

```
~]# yum -v grouplist kde\*
Loading "product-id" plugin
Loading "refresh-packagekit" plugin
Loading "subscription-manager" plugin
Updating Red Hat repositories.
INFO:rhsm-app.repolib:repos updated: 0
Config time: 0.123
Yum Version: 3.2.29
Setting up Group Process
Looking for repo options for [rhel]
rpmdb time: 0.001
group time: 1.291
Available Groups:
    KDE Desktop (kde-desktop)
Done
```

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

```
yum groupinstall group_name
```

You can also install by *groupid*:

```
yum groupinstall groupid
```

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

yum install @group

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

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

8.2.5. Removing Packages

Similarly to package installation, Yum allows you to uninstall (remove in **RPM** and Yum 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**:

yum 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:

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

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

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



WARNING

Yum is not able to remove a package without also removing packages which depend on it. This type of operation can only be performed by **RPM**, is not advised, and can potentially leave your system in a non-functioning state or cause applications to misbehave and/or crash. For further information, see [Section B.2.4, “Uninstalling”](#) in the **RPM** chapter.

Removing a Package Group

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

yum groupremove group

yum remove @group

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

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

**IMPORTANT**

When you tell **yum** to remove a package group, it will remove every package in that group, even if those packages are members of other package groups or dependencies of other installed packages. However, you can instruct **yum** to remove only those packages which are not required by any other packages or groups by adding the **groupremove_leaf_only=1** directive to the **[main]** section of the **/etc/yum.conf** configuration file. For more information on this directive, see [Section 8.4.1, “Setting \[main\] Options”](#).

8.3. WORKING WITH TRANSACTION HISTORY

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

8.3.1. Listing Transactions

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

yum history list

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

yum history list all

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

yum history list start_id..end_id

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

yum history list glob_expression

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

```
~]# yum history list 1..5
Loaded plugins: product-id, refresh-packagekit, subscription-manager
ID | Login user      | Date and time | Action(s) | Altered
---|---|---|---|---
5 | Jaromir ... <jhradilek> | 2011-07-29 15:33 | Install | 1
```

```

4 | Jaromir ... <jhradilek> | 2011-07-21 15:10 | Install | 1
3 | Jaromir ... <jhradilek> | 2011-07-16 15:27 | I, U | 73
2 | System <unset> | 2011-07-16 15:19 | Update | 1
1 | System <unset> | 2011-07-16 14:38 | Install | 1106
history list

```

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

- **ID** – an integer value that identifies a particular transaction.
- **Login user** – the name of the user whose login session was used to initiate a transaction. This information is typically presented in the **Full Name <username>** form. For transactions that were not issued by a user (such as an automatic system update), **System <unset>** is used instead.
- **Date and time** – the date and time when a transaction was issued.
- **Action(s)** – a list of actions that were performed during a transaction as described in [Table 8.1, "Possible values of the Action\(s\) field"](#).
- **Altered** – the number of packages that were affected by a transaction, possibly followed by additional information as described in [Table 8.2, "Possible values of the Altered field"](#).

Table 8.1. Possible values of the Action(s) field

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

Table 8.2. Possible values of the Altered field

Symbol	Description
<	Before the transaction finished, the rpmdb database was changed outside Yum.
>	After the transaction finished, the rpmdb database was changed outside Yum.
*	The transaction failed to finish.

Symbol	Description
#	The transaction finished successfully, but yum returned a non-zero exit code.
E	The transaction finished successfully, but an error or a warning was displayed.
P	The transaction finished successfully, but problems already existed in the rpmdb database.
s	The transaction finished successfully, but the --skip-broken command-line option was used and certain packages were skipped.

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

yum history summary

To display only transactions in a given range, type:

yum history summary start_id..end_id

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

yum history summary glob_expression

For instance, a summary of the transaction history displayed above would look like the following:

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

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

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

yum history package-list glob_expression

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

```
~]# yum history package-list subscription-manager*
Loaded plugins: product-id, refresh-packagekit, subscription-manager
ID | Action(s) | Package
-----
3 | Updated   | subscription-manager-0.95.11-1.el6.x86_64
3 | Update    |           0.95.17-1.el6_1.x86_64
3 | Updated   | subscription-manager-firstboot-0.95.11-1.el6.x86_64
3 | Update    |           0.95.17-1.el6_1.x86_64
3 | Updated   | subscription-manager-gnome-0.95.11-1.el6.x86_64
3 | Update    |           0.95.17-1.el6_1.x86_64
1 | Install   | subscription-manager-0.95.11-1.el6.x86_64
1 | Install   | subscription-manager-firstboot-0.95.11-1.el6.x86_64
1 | Install   | subscription-manager-gnome-0.95.11-1.el6.x86_64
history package-list
```

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

8.3.2. Examining Transactions

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

```
yum history summary id
```

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

```
yum history info id
```

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

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

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

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

```
Install screen-4.0.3-16.el6.x86_64
Install yum-plugin-security-1.1.30-17.el6.noarch
history info
```

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

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

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

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

For instance, for the first transaction in the previous example, the **yum history addon-info** command would provide the following output:

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

history addon-info
```

In this example, three types of information are available:

- **config-main** – global Yum options that were in use during the transaction. See [Section 8.4.1, “Setting \[main\] Options”](#) for information on how to change global options.
- **config-repos** – options for individual Yum repositories. See [Section 8.4.2, “Setting \[repository\] Options”](#) for information on how to change options for individual repositories.
- **saved_tx** – the data that can be used by the **yum load-transaction** command in order to repeat the transaction on another machine (see below).

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

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

8.3.3. Reverting and Repeating Transactions

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

```
yum history undo id
```

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

```
yum history redo id
```

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

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

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

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

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

```
yum load-transaction file_name
```

Note, however that the **rpmdb** version stored in the file must be identical to the version on the target system. You can verify the **rpmdb** version by using the **yum version nogroups** command.

8.3.4. Completing Transactions

An unexpected situation, such as power loss or system crash, can prevent you from completing your yum transaction. When such event occurs in the middle of your transaction, you can try to resume it later with the following command as **root**:

```
yum-complete-transaction
```

The **yum-complete-transaction** tool searches for incomplete or aborted yum transactions on a system and attempts to complete them. By default, these transactions are listed in the **/var/lib/yum/transaction-all** and **/var/lib/yum/transaction-done** files. If there are more unfinished transactions, **yum-complete-transaction** attempts to complete the most recent one first.

To clean transaction journal files without attempting to resume the aborted transactions, use the **--cleanup-only** option:

```
yum-complete-transaction --cleanup-only
```

8.3.5. Starting New Transaction History

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

```
yum history new
```

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

8.4. CONFIGURING YUM AND YUM REPOSITORIES

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

This section shows you how to:

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

8.4.1. Setting **[main]** Options

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

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

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

[comments abridged]

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

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

assumeyes=value

where *value* is one of:

0 – **yum** should prompt for confirmation of critical actions it performs. This is the default.

1 – Do not prompt for confirmation of critical **yum** actions. If **assumeyes=1** is set, **yum** behaves in the same way that the command-line option **-y** does.

cachedir=directory

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

See [Section 8.4.3, "Using Yum Variables"](#) for descriptions of the **\$basearch** and **\$releasever** Yum variables.

debuglevel=value

where *value* is an integer between **1** and **10**. Setting a higher **debuglevel** value causes **yum** to display more detailed debugging output. **debuglevel=0** disables debugging output, while **debuglevel=2** is the default.

exactarch=value

where *value* is one of:

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

1 – Consider the exact architecture when updating packages. With this setting, **yum** will not install an i686 package to update an i386 package already installed on the system. This is the default.

exclude=package_name [more_package_names]

This option allows you to exclude packages by keyword during installation/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/yum.conf** file, it sets the GPG-checking rule for all repositories. However, you can also set **gpgcheck=value** for individual repositories instead; that is, you can enable GPG-checking on one repository while disabling it on another. Setting **gpgcheck=value** for an individual repository in its corresponding **.repo** file overrides the default if it is present in **/etc/yum.conf**.

For more information on GPG signature-checking, see [Section B.3, "Checking a Package's Signature"](#).

groupremove_leaf_only=value

where *value* is one of:

0 – **yum** should *not* check the dependencies of each package when removing a package group. With this setting, **yum** removes all packages in a package group, regardless of whether those packages are required by other packages or groups. **groupremove_leaf_only=0** is the default.

1 – **yum** should check the dependencies of each package when removing a package group, and remove only those packages which are not required by any other package or group.

For more information on removing packages, see [Intelligent package group removal](#).

installonlypkgs=space separated list of packages

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

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

installonly_limit=value

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/yum.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.

logfile=file_name

where *file_name* is an absolute path to the file in which **yum** should write its logging output. By default, **yum** logs to **/var/log/yum.log**.

multilib_policy=value

where *value* is one of:

best – install the best-choice architecture for this system. For example, setting **multilib_policy=best** on an AMD64 system causes **yum** to install 64-bit versions of all packages.

all – always install every possible architecture for every package. For example, with **multilib_policy** set to **all** on an AMD64 system, **yum** would install both the i686 and AMD64 versions of a package, if both were available.

obsoletes=value

where *value* is one of:

0 – Disable **yum**'s obsoletes processing logic when performing updates.

1 – Enable **yum**'s obsoletes processing logic when performing updates. When one package declares in its spec file that it *obsoletes* another package, the latter package will be replaced by the former package when the former package is installed. Obsoletes are declared, for example, when a package is renamed. **obsoletes=1** the default.

plugins=value

where *value* is one of:

0 – Disable all Yum plug-ins globally.

**IMPORTANT**

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

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

For more information about various Yum plug-ins, see [Section 8.5, “Yum Plug-ins”](#). For further information on controlling plug-ins, see [Section 8.5.1, “Enabling, Configuring, and Disabling Yum Plug-ins”](#).

reposdir=directory

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

retries=value

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

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

8.4.2. Setting [repository] Options

The **[repository]** sections, where *repository* is a unique repository ID such as **my_personal_repo** (spaces are not permitted), allow you to define individual Yum repositories. To avoid conflicts, custom repositories should not use names used by Red Hat 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.

baseurl=repository_url

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

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

Usually this URL is an HTTP link, such as:

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

Note that Yum always expands the **\$releasever**, **\$arch**, and **\$basearch** variables in URLs. For more information about Yum variables, see [Section 8.4.3, “Using Yum Variables”](#).

Another useful **[repository]** directive is the following:

enabled=value

where *value* is one of:

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

1 – Include this repository as a package source.

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

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

Example 8.6. A sample /etc/yum.repos.d/redhat.repo file

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

```
#  
# Red Hat Repositories  
# Managed by (rhsmd) subscription-manager  
  
[red-hat-enterprise-linux-scalable-file-system-for-rhel-6-entitlement-rpms]  
name = Red Hat Enterprise Linux Scalable File System (for RHEL 6 Entitlement) (RPMS)  
baseurl = https://cdn.redhat.com/content/dist/rhel/entitlement-  
6/releases/$releasever/$basearch/scalablefilesystem/os  
enabled = 1
```

```

gpgcheck = 1
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
sslverify = 1
sslcacert = /etc/rhsm/ca/redhat-uep.pem
sslclientkey = /etc/pki/entitlement/key.pem
sslclientcert = /etc/pki/entitlement/11300387955690106.pem

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

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

```

8.4.3. Using Yum Variables

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

\$releasever

You can use this variable to reference the release version of Red Hat Enterprise Linux. Yum obtains the value of **\$releasever** from the **distroverpkg=value** line in the **/etc/yum.conf** configuration file. If there is no such line in **/etc/yum.conf**, then **yum** infers the correct value by deriving the version number from the redhat-release-server package. The value of **\$releasever** typically consists of the major release number and the variant of Red Hat Enterprise Linux, for example **6Client**, or **6Server**.

\$arch

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

\$basearch

You can use **\$basearch** to reference the base architecture of the system. For example, i686 machines have a base architecture of **i386**, and AMD64 and Intel 64 machines have a base architecture of **x86_64**.

\$YUM0-9

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

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

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

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

Instead of “Red Hat Enterprise Linux 6”, you can now use the following in the `.repo` files:

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

8.4.4. Viewing the Current Configuration

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

```
yum-config-manager
```

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

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

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

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

For example, to list all configuration options and their corresponding values, type the following at a shell prompt:

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

8.4.5. Adding, Enabling, and Disabling a Yum Repository

Section 8.4.2, “Setting [repository] Options” described various options you can use to define a Yum repository. This section explains how to add, enable, and disable a repository by using the **yum-config-manager** command.



IMPORTANT

When the system is registered with the certificate-based **Red Hat Network**, the Red Hat Subscription Manager tools are used to manage repositories in the **/etc/yum.repos.d/redhat.repo** file. See [Chapter 6, Registering the System and Managing Subscriptions](#) for more information how to register a system with **Red Hat Network** and use the Red Hat Subscription Manager tools to manage subscriptions.

Adding a Yum Repository

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



WARNING

Obtaining and installing software packages from unverified or untrusted software sources other than Red Hat Network constitutes a potential security risk, and could lead to security, stability, compatibility, and maintainability issues.

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

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

where *repository_url* is a link to the **.repo** file. For example, to add a repository located at <http://www.example.com/example.repo>, type the following at a shell prompt:

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

Enabling a Yum Repository

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

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

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

yum-config-manager --enable *glob_expression*

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

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

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

Disabling a Yum Repository

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

yum-config-manager --disable *repository*

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

yum-config-manager --disable *glob_expression*

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

8.4.6. Creating a Yum Repository

To set up a Yum repository, follow these steps:

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

```
yum install createrepo
```

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

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

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

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



IMPORTANT

Compared to Red Hat Enterprise Linux 5, RPM packages for Red Hat Enterprise Linux 6 are compressed with the XZ lossless data compression format and can be signed with newer hash algorithms like SHA-256. Consequently, it is not recommended to use the **createrepo** command on Red Hat Enterprise Linux 5 to create the package metadata for Red Hat Enterprise Linux 6.

8.4.7. Working with Yum Cache

By default, **yum** deletes downloaded data files when they are no longer needed after a successful operation. This minimizes the amount of storage space that **yum** uses. However, you can enable caching, so that the package files downloaded by **yum** stay in cache directories. By using cached data, you can carry out certain operations without a network connection, you can also copy packages stored in the caches and reuse them elsewhere.

Yum stores temporary files in the **/var/cache/yum/\$basearch/\$releasever/** directory, where **\$basearch** and **\$releasever** are Yum variables referring to base architecture of the system and the release version of Red Hat Enterprise Linux. Each configured repository has one subdirectory. For example, the directory **/var/cache/yum/\$basearch/\$releasever/development/packages/** holds packages downloaded from the development repository. You can find the values for the **\$basearch** and **\$releasever** variables in the output of the **yum version** command.

To change the default cache location, modify the **cachedir** option in the **[main]** section of the **/etc/yum.conf** configuration file. See [Section 8.4, “Configuring Yum and Yum Repositories”](#) for more information on configuring **yum**.

Enabling the Caches

To retain the cache of packages after a successful installation, add the following text to the **[main]** section of **/etc/yum.conf**.

```
keepcache = 1
```

Once you enabled caching, every yum operation may download package data from the configured repositories.

To download and make usable all the metadata for the currently enabled yum repositories, type:

```
yum makecache
```

This is useful if you want to make sure that the cache is fully up to date with all metadata. To set the time after which the metadata will expire, use the **metadata-expire** setting in **/etc/yum.conf**.

Using yum in Cache-only Mode

To carry out a **yum** command without a network connection, add the **-C** or **--cacheonly** command-line option. With this option, **yum** proceeds without checking any network repositories, and uses only cached files. In this mode, **yum** may only install packages that have been downloaded and cached by a previous operation.

For instance, to list packages that use the currently cached data with names that contain “gstreamer”, enter the following command:

```
yum -C list gstreamer*
```

Clearing the yum Caches

It is often useful to remove entries accumulated in the `/var/cache/yum` directory. If you remove a package from the cache, you do not affect the copy of the software installed on your system. To remove all entries for currently enabled repositories from the cache, type the following as a **root**:

yum clean all

There are various ways to invoke **yum** in **clean** mode depending on the type of cached data you want to remove. See [Table 8.3, “Available yum clean options”](#) for a complete list of available configuration options.

Table 8.3. Available yum clean options

Option	Description
expire-cache	eliminates time records of the metadata and mirrorlists download for each repository. This forces yum to revalidate the cache for each repository the next time it is used.
packages	eliminates any cached packages from the system
headers	eliminates all header files that previous versions of yum used for dependency resolution
metadata	eliminates all files that yum uses to determine the remote availability of packages. These metadata are downloaded again the next time yum is run.
dbcache	eliminates the sqlite cache used for faster access to metadata. Using this option will force yum to download the sqlite metadata the next time it is run. This does not apply for repositories that contain only .xml data, in that case, sqlite data are deleted but without subsequent download
rpmbdb	eliminates any cached data from the local rpmbdb
plugins	enabled plugins are forced to eliminate their cached data
all	removes all of the above

The **expire-cache** option is most preferred from the above list. In many cases, it is a sufficient and much faster replacement for **clean all**.

8.4.8. Adding the Optional and Supplementary Repositories

Optional and Supplementary subscription channels provide additional software packages for Red Hat Enterprise Linux that cover open source licensed software (in the Optional channel) and proprietary licensed software (in the Supplementary channel).

Before subscribing to the Optional and Supplementary channels see the [Scope of Coverage Details](#). If you decide to install packages from these channels, follow the steps documented in the article called [How to access Optional and Supplementary channels, and -devel packages using Red Hat Subscription Manager \(RHSM\)?](#) on the Red Hat Customer Portal.

8.5. YUM PLUG-INS

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

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

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

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

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

```
plugins=1
```

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



IMPORTANT

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

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

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

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

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

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

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

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

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

```
~]# yum update --disableplugin=presto,refresh-pack*
```

8.5.2. Installing Additional Yum Plug-ins

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

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

8.5.3. Plug-in Descriptions

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

search-disabled-repos (subscription-manager)

The **search-disabled-repos** plug-in allows you to temporarily or permanently enable disabled repositories to help resolve dependencies. With this plug-in enabled, when Yum fails to install a package due to failed dependency resolution, it offers to temporarily enable disabled repositories and try again. If the installation succeeds, Yum also offers to enable the used repositories permanently. Note that the plug-in works only with the repositories that are managed by **subscription-manager** and not with custom repositories.



IMPORTANT

If **yum** is executed with the **--assumeyes** or **-y** option, or if the **assumeyes** directive is enabled in **/etc/yum.conf**, the plug-in enables disabled repositories, both temporarily and permanently, without prompting for confirmation. This may lead to problems, for example, enabling repositories that you do not want enabled.

To configure the **search-disabled-repos** plug-in, edit the configuration file located in **/etc/yum/pluginconf.d/search-disabled-repos.conf**. For the list of directives you can use in the **[main]** section, see the table below.

Table 8.4. Supported **search-disabled-repos.conf** directives

Directive	Description
-----------	-------------

Directive	Description
enabled =value	Allows you to enable or disable the plug-in. The <i>value</i> must be either 1 (enabled), or 0 (disabled). The plug-in is enabled by default.
notify_only =value	Allows you to restrict the behavior of the plug-in to notifications only. The <i>value</i> must be either 1 (notify only without modifying the behavior of Yum), or 0 (modify the behavior of Yum). By default the plug-in only notifies the user.
ignored_repos =repositories	Allows you to specify the repositories that will not be enabled by the plug-in.

kabi (kabi-yum-plugins)

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

To configure the **kabi** plug-in, edit the configuration file located in `/etc/yum/pluginconf.d/kabi.conf`. See [Table 8.5, "Supported kabi.conf directives"](#) for a list of directives that can be used in the **[main]** section.

Table 8.5. Supported kabi.conf directives

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

presto (yum-presto)

The **presto** plug-in adds support to Yum for downloading *delta RPM* packages, during updates, from repositories which have **presto** metadata enabled. Delta RPMs contain only the differences between the version of the package installed on the client requesting the RPM package and the updated version in the repository.

Downloading a delta RPM is much quicker than downloading the entire updated package, and can speed up updates considerably. Once the delta RPMs are downloaded, they must be rebuilt to apply the difference to the currently-installed package and thus create the full, updated package. This

process takes CPU time on the installing machine. Using delta RPMs is therefore a compromise between time-to-download, which depends on the network connection, and time-to-rebuild, which is CPU-bound. Using the **presto** plug-in is recommended for fast machines and systems with slower network connections, while slower machines on very fast connections benefit more from downloading normal RPM packages, that is, by disabling **presto**.

product-id (subscription-manager)

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

refresh-packagekit (PackageKit-yum-plugin)

The **refresh-packagekit** plug-in updates metadata for **PackageKit** whenever **yum** is run. The **refresh-packagekit** plug-in is installed by default.

rhnplugin (yum-rhn-plugin)

The **rhnplugin** provides support for connecting to **RHN Classic**. This allows systems registered with **RHN Classic** to update and install packages from this system. Note that **RHN Classic** is only provided for older Red Hat Enterprise Linux systems (that is, Red Hat Enterprise Linux 4.x, Red Hat Enterprise Linux 5.x, and Satellite 5.x) in order to migrate them over to Red Hat Enterprise Linux 6. The **rhnplugin** is installed by default.

See the **rhnplugin(8)** manual page for more information about the plug-in.

security (yum-plugin-security)

Discovering information about and applying security updates easily and often is important to all system administrators. For this reason Yum provides the **security** plug-in, which extends **yum** with a set of highly-useful security-related commands, subcommands and options.

You can check for security-related updates as follows:

```
~]# yum check-update --security
Loaded plugins: product-id, refresh-packagekit, security, subscription-manager
Updating Red Hat repositories.
INFO:rhsm-app.repolib:repos updated: 0
Limiting package lists to security relevant ones
Needed 3 of 7 packages, for security
elinks.x86_64          0.12-0.13.el6      rhel
kernel.x86_64          2.6.30.8-64.el6    rhel
kernel-headers.x86_64   2.6.30.8-64.el6    rhel
```

You can then use either **yum update --security** or **yum update-minimal --security** to update those packages which are affected by security advisories. Both of these commands update all packages on the system for which a security advisory has been issued. **yum update-minimal --security** updates them to the latest packages which were released as part of a security advisory, while **yum update --security** will update all packages affected by a security advisory *to the latest version of that package available*.

In other words, if:

- the kernel-2.6.30.8-16 package is installed on your system;
- the kernel-2.6.30.8-32 package was released as a security update;
- then kernel-2.6.30.8-64 was released as a bug fix update,

...then **yum update-minimal --security** will update you to kernel-2.6.30.8-32, and **yum update --security** will update you to kernel-2.6.30.8-64. Conservative system administrators probably want to use **update-minimal** to reduce the risk incurred by updating packages as much as possible.

See the **yum-security(8)** manual page for usage details and further explanation of the enhancements the **security** plug-in adds to **yum**.

subscription-manager (subscription-manager)

The **subscription-manager** plug-in provides support for connecting to **Red Hat Network**. This allows systems registered with **Red Hat Network** to update and install packages from the certificate-based Content Delivery Network. The **subscription-manager** plug-in is installed by default.

See [Chapter 6, Registering the System and Managing Subscriptions](#) for more information how to manage product subscriptions and entitlements.

yum-downloadonly (yum-plugin-downloadonly)

The **yum-downloadonly** plug-in provides the **--downloadonly** command-line option which can be used to download packages from Red Hat Network or a configured Yum repository without installing the packages.

To install the package, follow the instructions in [Section 8.5.2, “Installing Additional Yum Plug-ins”](#). After the installation, see the contents of the **/etc/yum/pluginconf.d/downloadonly.conf** file to ensure that the plug-in is enabled:

```
~]$ cat /etc/yum/pluginconf.d/downloadonly.conf
[main]
enabled=1
```

In the following example, the **yum install --downloadonly** command is run to download the latest version of the httpd package, without installing it:

```
~]# yum install httpd --downloadonly
Loaded plugins: downloadonly, product-id, refresh-packagekit, rhnplugin,
               : subscription-manager
Updating Red Hat repositories.
Setting up Install Process
Resolving Dependencies
--> Running transaction check
--> Package httpd.x86_64 0:2.2.15-9.el6_1.2 will be updated
--> Package httpd.x86_64 0:2.2.15-15.el6_2.1 will be an update
--> Processing Dependency: httpd-tools = 2.2.15-15.el6_2.1 for package: httpd-2.2.15-
    15.el6_2.1.x86_64
--> Running transaction check
--> Package httpd-tools.x86_64 0:2.2.15-9.el6_1.2 will be updated
--> Package httpd-tools.x86_64 0:2.2.15-15.el6_2.1 will be an update
--> Finished Dependency Resolution

Dependencies Resolved
```

```
=====
=
Package      Arch      Version           Repository      Size
=====
```

```
=
Updating:
httpd      x86_64  2.2.15-15.el6_2.1    rhel-x86_64-server-6   812 k
Updating for dependencies:
httpd-tools x86_64  2.2.15-15.el6_2.1    rhel-x86_64-server-6   70 k

Transaction Summary
=====
=
Upgrade     2 Package(s)

Total download size: 882 k
Is this ok [y/N]: y
Downloading Packages:
(1/2): httpd-2.2.15-15.el6_2.1.x86_64.rpm           | 812 kB  00:00
(2/2): httpd-tools-2.2.15-15.el6_2.1.x86_64.rpm       | 70 kB   00:00
-----
Total                                         301 kB/s | 882 kB  00:02
```

exiting because --downloadonly specified

By default, packages downloaded using the **--downloadonly** option are saved in one of the subdirectories of the **/var/cache/yum** directory, depending on the Red Hat Enterprise Linux variant and architecture.

If you want to specify an alternate directory to save the packages, pass the **--downloaddir** option along with **--downloadonly**:

```
~]# yum install --downloadonly --downloaddir=/path/to/directory httpd
```

NOTE

As an alternative to the **yum-downloadonly** plugin – to download packages without installing them – you can use the **yumdownloader** utility that is provided by the **yum-utils** package.

8.6. ADDITIONAL RESOURCES

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

Installed Documentation

- **yum(8)** – The manual page for the **yum** command-line utility provides a complete list of supported options and commands.
- **yumdb(8)** – The manual page for the **yumdb** command-line utility documents how to use this tool to query and, if necessary, alter the yum database.
- **yum.conf(5)** – The manual page named **yum.conf** documents available yum configuration options.

- **yum-utils(1)** – The manual page named **yum-utils** lists and briefly describes additional utilities for managing yum configuration, manipulating repositories, and working with yum database.

Online Resources

- [Yum Guides](#) – The *Yum Guides* page on the project home page provides links to further documentation.
- [Red Hat Access Labs](#) – The Red Hat Access Labs includes a “Yum Repository Configuration Helper”.

See Also

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

CHAPTER 9. PACKAGEKIT

Red Hat provides **PackageKit** for viewing, managing, updating, installing and uninstalling packages compatible with your system. PackageKit consists of several graphical interfaces that can be opened from the GNOME panel menu, or from the Notification Area when PackageKit alerts you that updates are available. For more information on **PackageKit's** architecture and available front ends, see [Section 9.3, “PackageKit Architecture”](#).

9.1. UPDATING PACKAGES WITH SOFTWARE UPDATE

PackageKit displays a starburst icon in the Notification Area whenever updates are available to be installed on your system.

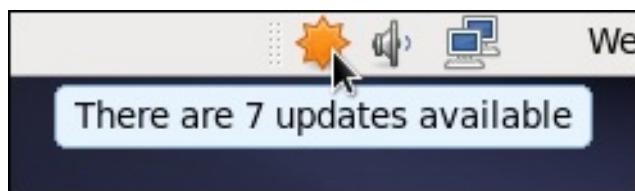


Figure 9.1. PackageKit's icon in the Notification Area

Clicking on the notification icon opens the **Software Update** window. Alternatively, you can open **Software Updates** by clicking **System → Administration → Software Update** from the GNOME panel, or running the **gpk-update-viewer** command at the shell prompt. In the **Software Updates** window, all available updates are listed along with the names of the packages being updated (minus the **.rpm** suffix, but including the CPU architecture), a short summary of the package, and, usually, short descriptions of the changes the update provides. Any updates you do not want to install can be de-selected here by unchecking the check box corresponding to the update.

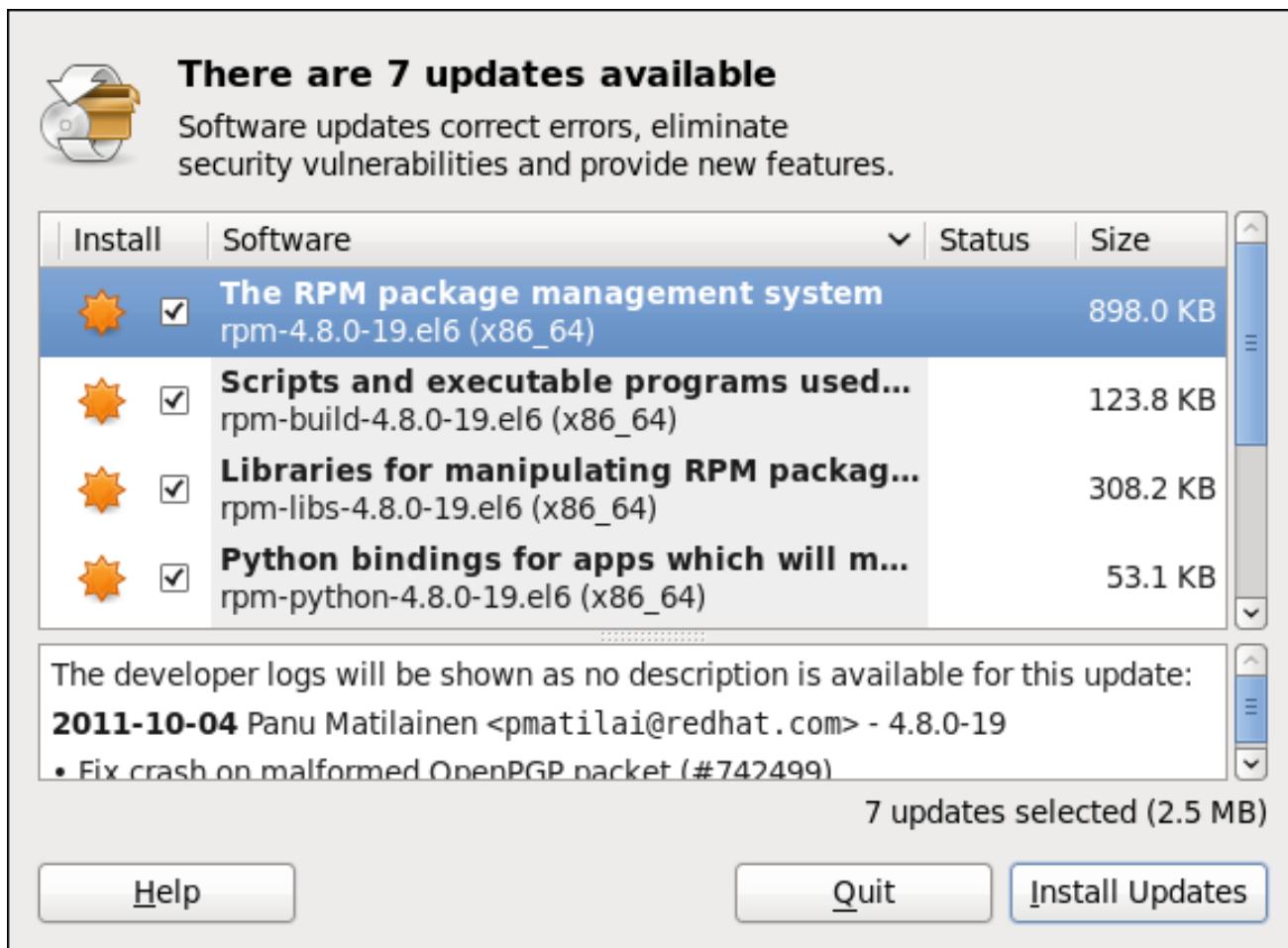


Figure 9.2. Installing updates with Software Update

The updates presented in the **Software Updates** window only represent the currently-installed packages on your system for which updates are available; dependencies of those packages, whether they are existing packages on your system or new ones, are not shown until you click **Install Updates**.

PackageKit utilizes the fine-grained user authentication capabilities provided by the **PolicyKit** toolkit whenever you request it to make changes to the system. Whenever you instruct PackageKit to update, install or remove packages, you will be prompted to enter the superuser password before changes are made to the system.

If you instruct PackageKit to update the kernel package, then it will prompt you after installation, asking you whether you want to reboot the system and thereby boot into the newly-installed kernel.

Setting the Update-Checking Interval

Right-clicking on PackageKit's Notification Area icon and clicking **Preferences** opens the **Software Update Preferences** window, where you can define the interval at which PackageKit checks for package updates, as well as whether or not to automatically install all updates or only security updates. Leaving the **Check for updates when using mobile broadband** box unchecked is handy for avoiding extraneous bandwidth usage when using a wireless connection on which you are charged for the amount of data you download.

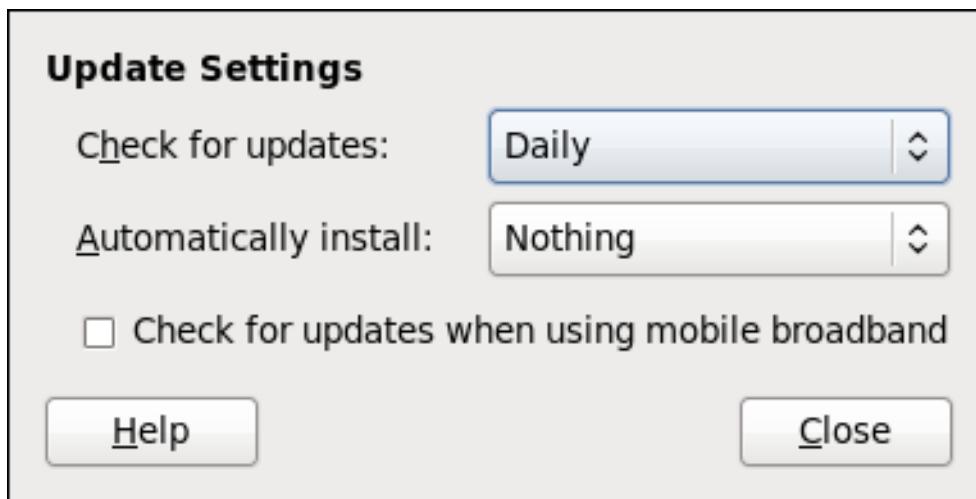


Figure 9.3. Setting PackageKit's update-checking interval

9.2. USING ADD/REMOVE SOFTWARE

To find and install a new package, on the GNOME panel click on **System → Administration → Add/Remove Software**, or run the **gpk-application** command at the shell prompt.

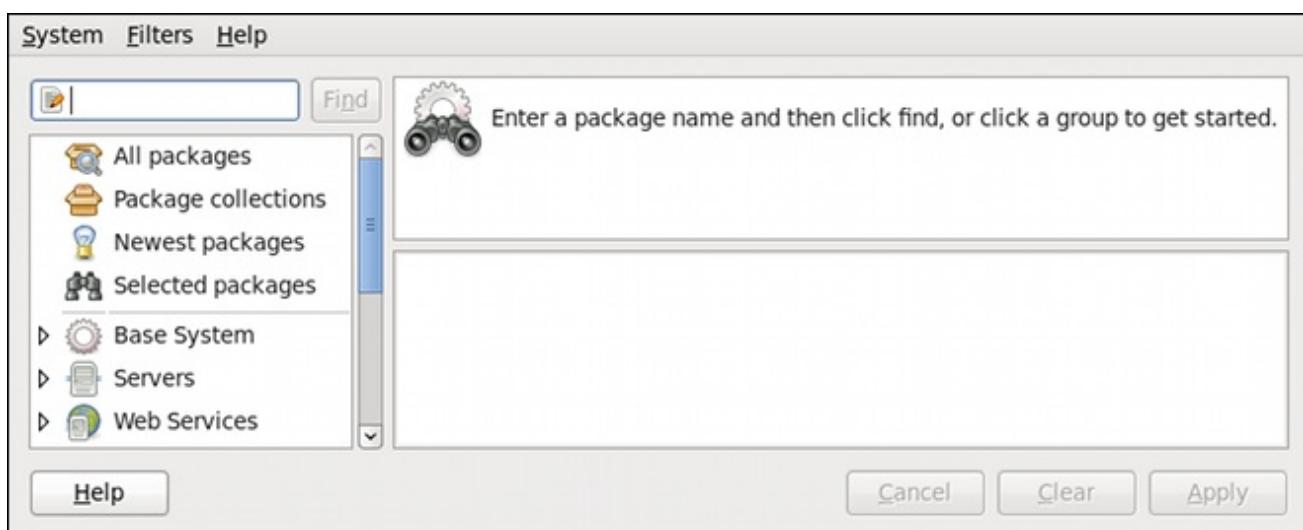


Figure 9.4. PackageKit's Add/Remove Software window

9.2.1. Refreshing Software Sources (Yum Repositories)

PackageKit refers to **Yum** repositories as software sources. It obtains all packages from enabled software sources. You can view the list of all *configured* and unfiltered (see below) Yum repositories by opening **Add/Remove Software** and clicking **System → Software sources**. The **Software Sources** dialog shows the repository name, as written on the **name=<My Repository Name>** field of all **[repository]** sections in the **/etc/yum.conf** configuration file, and in all **repository.repo** files in the **/etc/yum.repos.d/** directory.

Entries which are checked in the **Enabled** column indicate that the corresponding repository will be used to locate packages to satisfy all update and installation requests (including dependency resolution). You can enable or disable any of the listed Yum repositories by selecting or clearing the check box. Note that doing so causes **PolicyKit** to prompt you for superuser authentication.

The **Enabled** column corresponds to the **enabled=<1 or 0>** field in **[repository]** sections. When you click the check box, PackageKit inserts the **enabled=<1 or 0>** line into the correct **[repository]** section if it

does not exist, or changes the value if it does. This means that enabling or disabling a repository through the **Software Sources** window causes that change to persist after closing the window or rebooting the system.

Note that it is not possible to add or remove Yum repositories through PackageKit.



NOTE

Checking the box at the bottom of the **Software Sources** window causes PackageKit to display source RPM, testing and debuginfo repositories as well. This box is unchecked by default.

After making a change to the available Yum repositories, click on **System → Refresh package lists** to make sure your package list is up-to-date.

9.2.2. Finding Packages with Filters

Once the software sources have been updated, it is often beneficial to apply some filters so that PackageKit retrieves the results of our **Find** queries faster. This is especially helpful when performing many package searches. Four of the filters in the **Filters** drop-down menu are used to split results by matching or not matching a single criterion. By default when PackageKit starts, these filters are all unapplied (**No filter**), but once you do filter by one of them, that filter remains set until you either change it or close PackageKit.

Because you are usually searching for available packages that are *not* installed on the system, click **Filters → Installed** and select the **Only available** radio button.

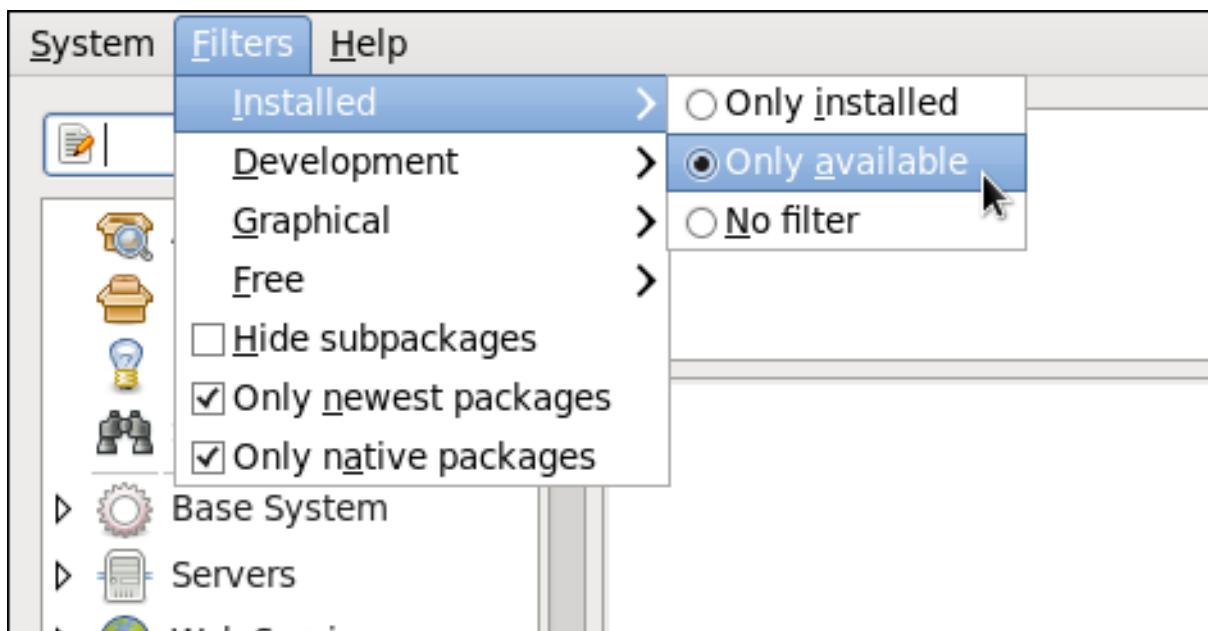


Figure 9.5. Filtering out already-installed packages

Also, unless you require development files such as C header files, click **Filters → Development** and select the **Only end user files** radio button. This filters out all of the **<package_name>-devel** packages we are not interested in.

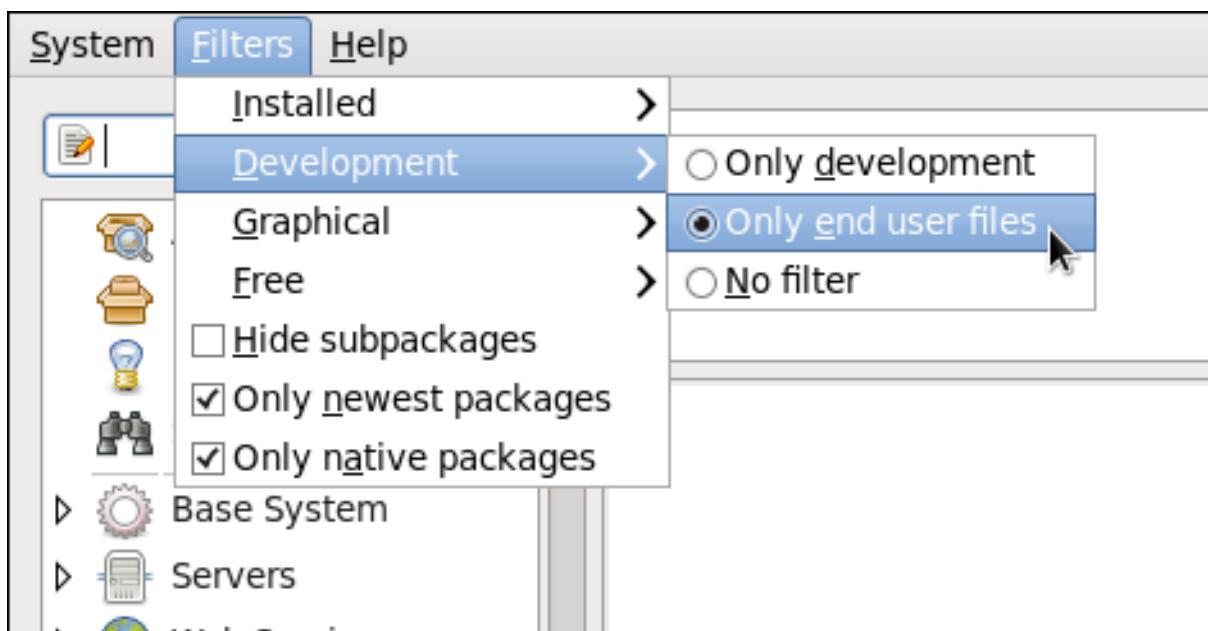


Figure 9.6. Filtering out development packages from the list of Find results

The two remaining filters with submenus are:

Graphical

Narrows the search to either applications which provide a GUI interface (**Only graphical**) or those that do not. This filter is useful when browsing for GUI applications that perform a specific function.

Free

Search for packages which are considered to be free software. See the [Fedora Licensing List](#) for details on approved licenses.

The remaining filters can be enabled by selecting the check boxes next to them:

Hide subpackages

Checking the **Hide subpackages** check box filters out generally-uninteresting packages that are typically only dependencies of other packages that we want. For example, checking **Hide subpackages** and searching for `<package>` would cause the following related packages to be filtered out of the **Find** results (if it exists):

- `<package>-devel`
- `<package>-libs`
- `<package>-libs-devel`
- `<package>-debuginfo`

Only newest packages

Checking **Only newest packages** filters out all older versions of the same package from the list of results, which is generally what we want. Note that this filter is often combined with the **Only available** filter to search for the latest available versions of new (not installed) packages.

Only native packages

Checking the **Only native packages** box on a multilib system causes PackageKit to omit listing

results for packages compiled for the architecture that runs in *compatibility mode*. For example, enabling this filter on a 64-bit system with an AMD64 CPU would cause all packages built for the 32-bit x86 CPU architecture not to be shown in the list of results, even though those packages are able to run on an AMD64 machine. Packages which are architecture-agnostic (i.e. `noarch` packages such as `crontabs-1.10-32.1.el6.noarch.rpm`) are never filtered out by checking **Only native packages**. This filter has no affect on non-multilib systems, such as x86 machines.

9.2.3. Installing and Removing Packages (and Dependencies)

With the two filters selected, **Only available** and **Only end user files**, search for the `screen` window manager for the command line and highlight the package. You now have access to some very useful information about it, including: a clickable link to the project homepage; the Yum package group it is found in, if any; the license of the package; a pointer to the GNOME menu location from where the application can be opened, if applicable; and the size of the package, which is relevant when we download and install it.

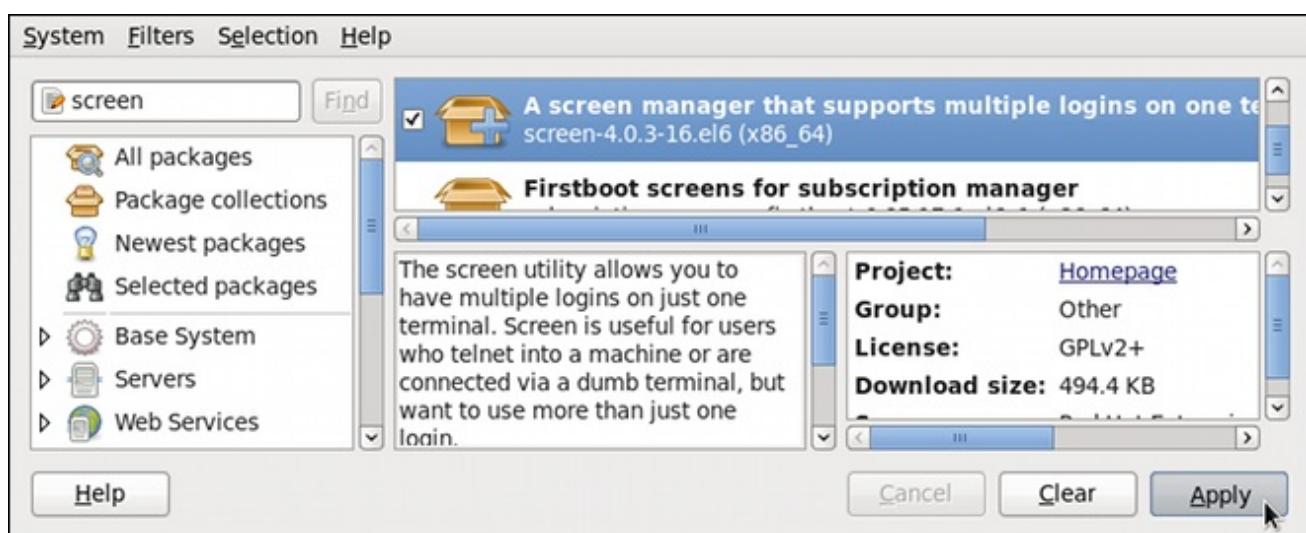


Figure 9.7. Viewing and installing a package with PackageKit's Add/Remove Software window

When the check box next to a package or group is checked, then that item is already installed on the system. Checking an unchecked box causes it to be *marked* for installation, which only occurs when the **Apply** button is clicked. In this way, you can search for and select multiple packages or package groups before performing the actual installation transactions. Additionally, you can remove installed packages by unchecking the checked box, and the removal will occur along with any pending installations when **Apply** is pressed. Dependency resolution, which may add additional packages to be installed or removed, is performed after pressing **Apply**. PackageKit will then display a window listing those additional packages to install or remove, and ask for confirmation to proceed.

Select `screen` and click the **Apply** button. You will then be prompted for the superuser password; enter it, and PackageKit will install `screen`. After finishing the installation, PackageKit sometimes presents you with a list of your newly-installed applications and offers you the choice of running them immediately. Alternatively, you will remember that finding a package and selecting it in the **Add/Remove Software** window shows you the **Location** of where in the GNOME menus its application shortcut is located, which is helpful when you want to run it.

Once it is installed, you can run `screen`, a screen manager that allows you to have multiple logins on one terminal, by typing `screen` at a shell prompt.

`screen` is a very useful utility, but we decide that we do not need it and we want to uninstall it. Remembering that we need to change the **Only available** filter we recently used to install it to **Only**

installed in **Filters → Installed**, we search for **screen** again and uncheck it. The program did not install any dependencies of its own; if it had, those would be automatically removed as well, as long as they were not also dependencies of any other packages still installed on our system.

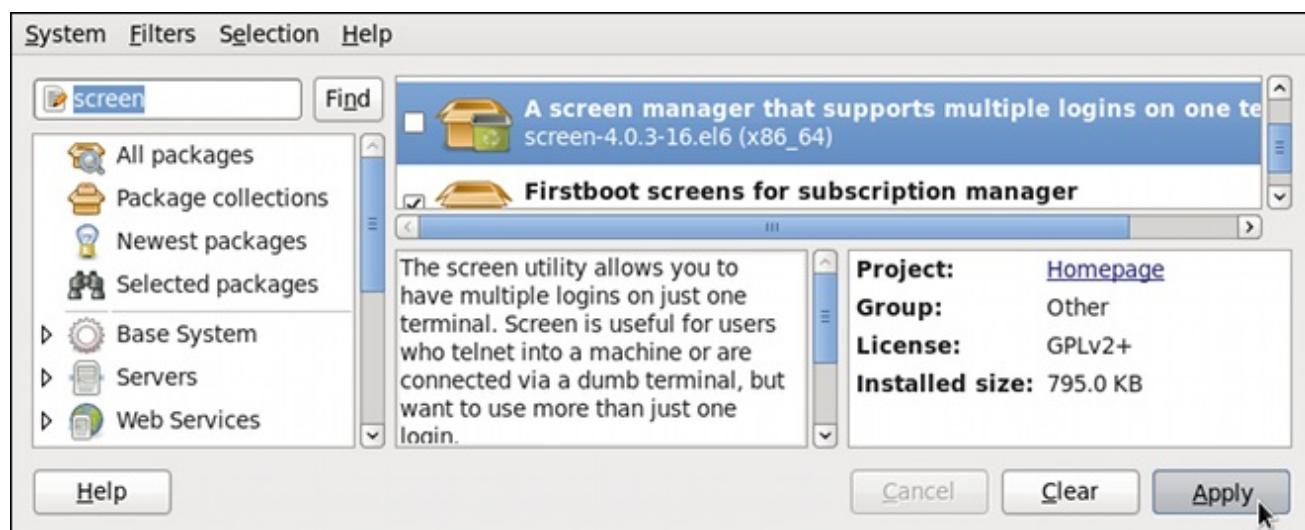
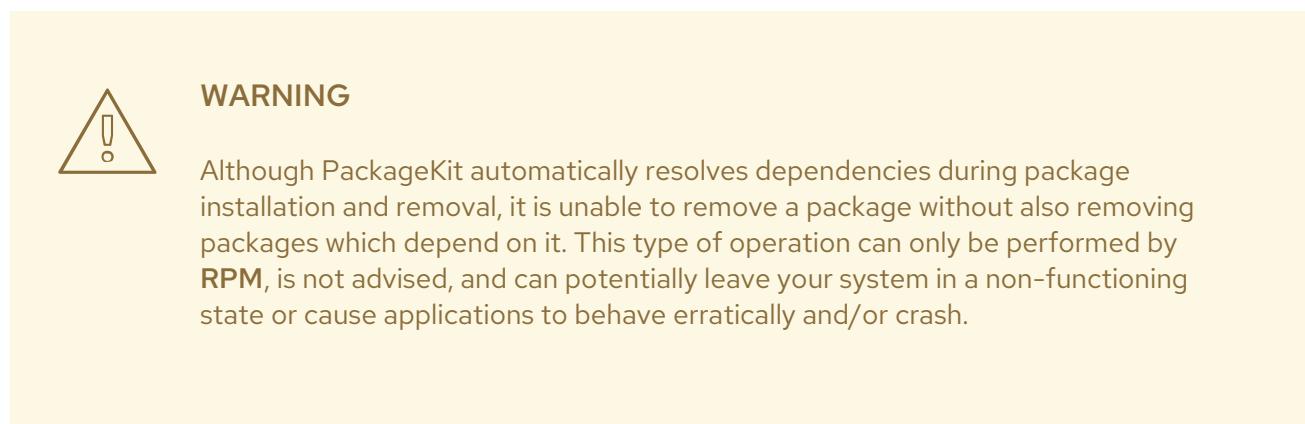


Figure 9.8. Removing a package with PackageKit's Add/Remove Software window

9.2.4. Installing and Removing Package Groups

PackageKit also has the ability to install Yum package groups, which it calls **Package collections**. Clicking on **Package collections** in the top-left list of categories in the **Software Updates** window allows us to scroll through and find the package group we want to install. In this case, we want to install Czech language support (the **Czech Support** group). Checking the box and clicking **apply** informs us how many *additional* packages must be installed in order to fulfill the dependencies of the package group.

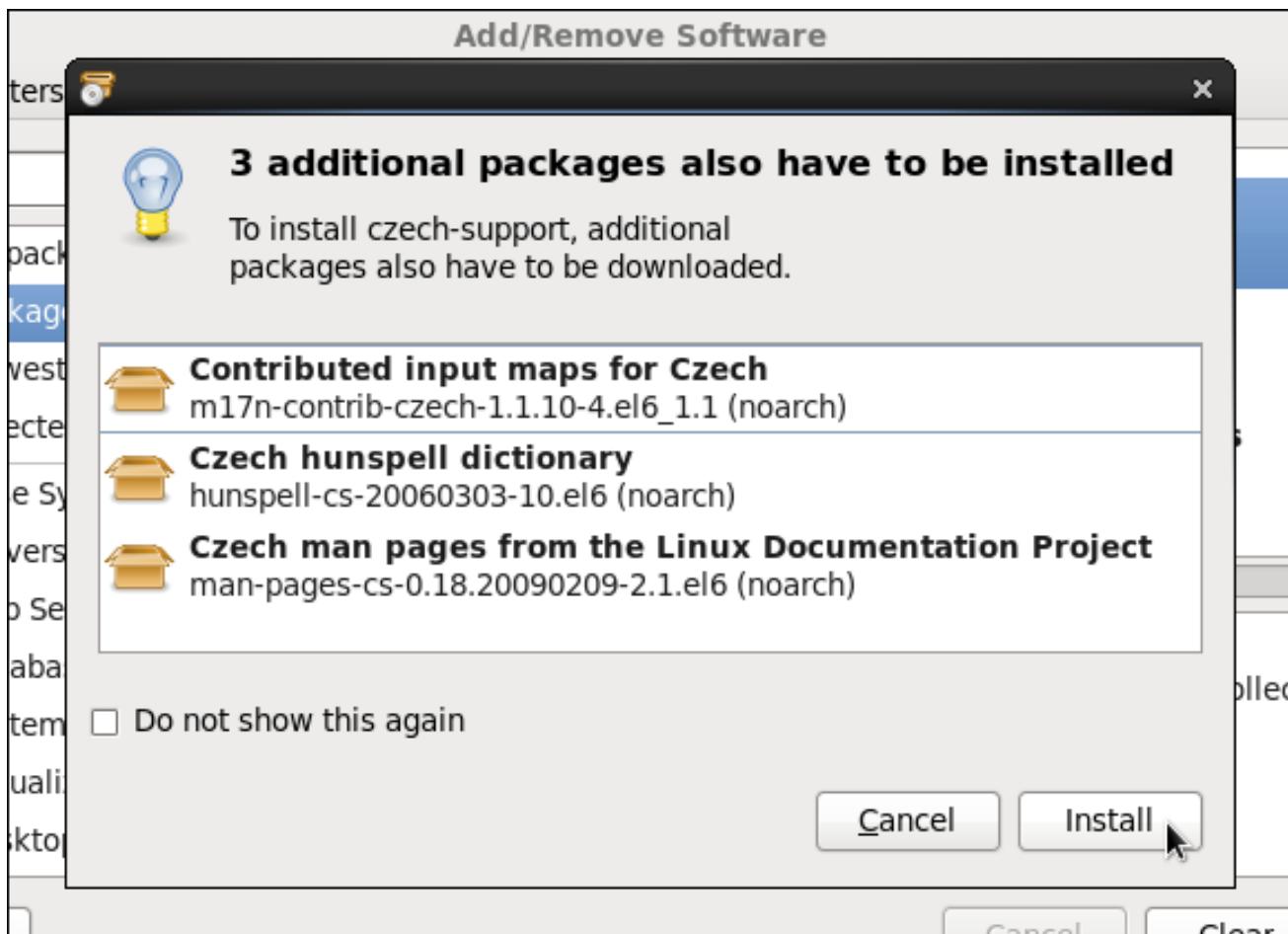


Figure 9.9. Installing the Czech Support package group

Similarly, installed package groups can be uninstalled by selecting **Package collections**, unchecking the appropriate check box, and applying.

9.2.5. Viewing the Transaction Log

PackageKit maintains a log of the transactions that it performs. To view the log, from the **Add/Remove Software** window, click **System → Software log**, or run the **gpk-log** command at the shell prompt.

The **Software Log Viewer** shows the following information:

- **Date** – the date on which the transaction was performed.
- **Action** – the action that was performed during the transaction, for example **Updated packages** or **Installed packages**.
- **Details** – the transaction type such as **Updated**, **Installed**, or **Removed**, followed by a list of affected packages.
- **Username** – the name of the user who performed the action.
- **Application** – the front end application that was used to perform the action, for example **Update System**.

Typing the name of a package in the top text entry field filters the list of transactions to those which affected that package.

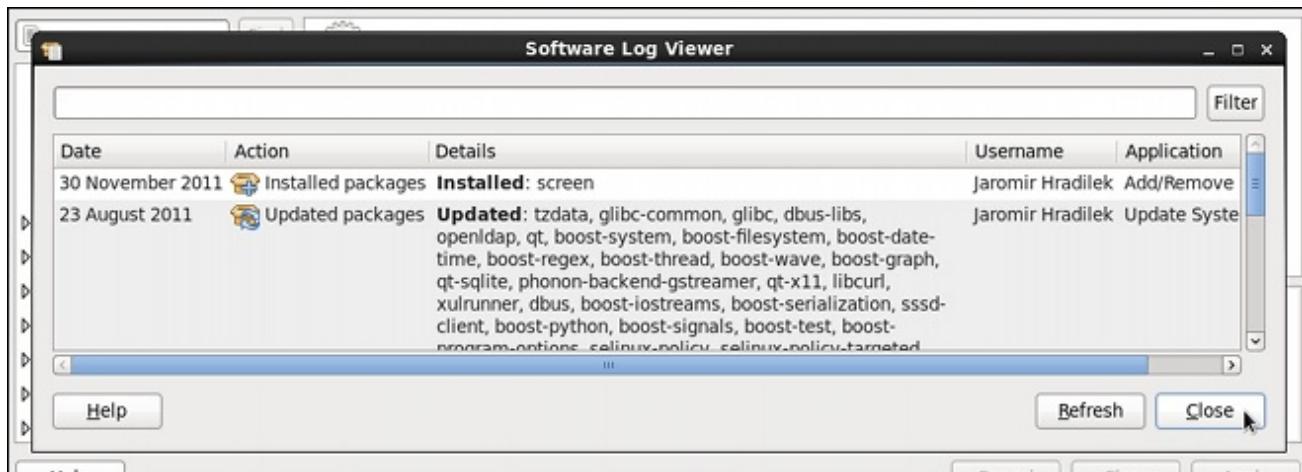


Figure 9.10. Viewing the log of package management transactions with the Software Log Viewer

9.3. PACKAGEKIT ARCHITECTURE

Red Hat provides the PackageKit suite of applications for viewing, updating, installing and uninstalling packages and package groups compatible with your system. Architecturally, PackageKit consists of several graphical front ends that communicate with the **packagekitd** daemon back end, which communicates with a package manager-specific back end that utilizes Yum to perform the actual transactions, such as installing and removing packages, etc.

[Table 9.1, “PackageKit GUI windows, menu locations, and shell prompt commands”](#) shows the name of the GUI window, how to start the window from the GNOME desktop or from the **Add/Remove Software** window, and the name of the command-line application that opens that window.

Table 9.1. PackageKit GUI windows, menu locations, and shell prompt commands

Window Title	Function	How to Open	Shell Command
Add/Remove Software	Install, remove or view package info	From the GNOME panel: System → Administration → Add/Remove Software	gpk-application
Software Update	Perform package updates	From the GNOME panel: System → Administration → Software Update	gpk-update-viewer
Software Sources	Enable and disable Yum repositories	From Add/Remove Software : System → Software Sources	gpk-repo
Software Log Viewer	View the transaction log	From Add/Remove Software : System → Software Log	gpk-log
Software Update Preferences	Set PackageKit preferences		gpk-prefs

Window Title	Function	How to Open	Shell Command
(Notification Area Alert)	Alerts you when updates are available	From the GNOME panel: System → Preferences → Startup Applications , the Startup Programs tab	<code>gpk-update-icon</code>

The **packagekitd** daemon runs outside the user session and communicates with the various graphical front ends. The **packagekitd** daemon^[2] communicates via the **DBus** system message bus with another back end, which utilizes Yum's Python API to perform queries and make changes to the system. On Linux systems other than Red Hat Enterprise Linux and Fedora, **packagekitd** can communicate with other back ends that are able to utilize the native package manager for that system. This modular architecture provides the abstraction necessary for the graphical interfaces to work with many different package managers to perform essentially the same types of package management tasks. Learning how to use the PackageKit front ends means that you can use the same familiar graphical interface across many different Linux distributions, even when they utilize a native package manager other than Yum.

In addition, PackageKit's separation of concerns provides reliability in that a crash of one of the GUI windows—or even the user's X Window session—will not affect any package management tasks being supervised by the **packagekitd** daemon, which runs outside of the user session.

All of the front end graphical applications discussed in this chapter are provided by the `gnome-packagekit` package instead of by PackageKit and its dependencies.

Finally, PackageKit also comes with a console-based front end called **pkcon**.

9.4. ADDITIONAL RESOURCES

For more information about PackageKit, see the resources listed below.

Installed Documentation

- **gpk-application(1)** – The manual page containing information about the **gpk-application** command.
- **gpk-backend-status(1)** – The manual page containing information about the **gpk-backend-status** command.
- **gpk-install-local-file(1)** – The manual page containing information about the **gpk-install-local-file** command.
- **gpk-install-mime-type(1)** – The manual page containing information about the **gpk-install-mime-type** command.
- **gpk-install-package-name(1)** – The manual page containing information about the **gpk-install-package-name** command.
- **gpk-install-package-name(1)** – The manual page containing information about the **gpk-install-package-name** command.
- **gpk-prefs(1)** – The manual page containing information about the **gpk-prefs** command.
- **gpk-repo(1)** – The manual page containing information about the **gpk-repo** command.

- **gpk-update-icon(1)** – The manual page containing information about the **gpk-update-icon** command.
- **gpk-update-viewer(1)** – The manual page containing information about the **gpk-update-viewer** command.
- **pkcon(1)** and **pkmon(1)** – The manual pages containing information about the PackageKit console client.
- **pkgenpack(1)** – The manual page containing information about the PackageKit Pack Generator.

Online Documentation

- [PackageKit home page](#) – The PackageKit home page listing detailed information about the PackageKit software suite.
- [PackageKit FAQ](#) – An informative list of Frequently Asked Questions for the PackageKit software suite.

See Also

- [Chapter 8, Yum](#) documents Yum, the Red Hat package manager.

[2] System daemons are typically long-running processes that provide services to the user or to other programs, and which are started, often at boot time, by special initialization scripts (often shortened to *init scripts*). Daemons respond to the **service** command and can be turned on or off permanently by using the **chkconfig on** or **chkconfig off** commands. They can typically be recognized by a “d” appended to their name, such as the **packagekitd** daemon. See [Chapter 12, Services and Daemons](#) for information about system services.

PART IV. NETWORKING

This part describes how to configure the network on Red Hat Enterprise Linux.

CHAPTER 10. NETWORKMANAGER

NetworkManager is a dynamic network control and configuration system that attempts to keep network devices and connections up and active when they are available. **NetworkManager** consists of a core daemon, a GNOME Notification Area applet that provides network status information, and graphical configuration tools that can create, edit and remove connections and interfaces.

NetworkManager can be used to configure the following types of connections: Ethernet, wireless, mobile broadband (such as cellular 3G), and **DSL** and **PPPoE** (Point-to-Point over Ethernet). In addition, **NetworkManager** allows for the configuration of network aliases, static routes, DNS information and VPN connections, as well as many connection-specific parameters. Finally, **NetworkManager** provides a rich API via D-Bus which allows applications to query and control network configuration and state.

Previous versions of Red Hat Enterprise Linux included the **Network Administration Tool**, which was commonly known as **system-config-network** after its command-line invocation. In Red Hat Enterprise Linux 6, **NetworkManager** replaces the former **Network Administration Tool** while providing enhanced functionality, such as user-specific and mobile broadband configuration. It is also possible to configure the network in Red Hat Enterprise Linux 6 by editing interface configuration files; see [Chapter 11, Network Interfaces](#) for more information.

NetworkManager may be installed by default on your version of Red Hat Enterprise Linux. To ensure that it is, run the following command as **root**:

```
~]# yum install NetworkManager
```

10.1. THE NETWORKMANAGER DAEMON

The **NetworkManager** daemon runs with **root** privileges and is usually configured to start up at boot time. You can determine whether the **NetworkManager** daemon is running by entering this command as **root**:

```
~]# service NetworkManager status  
NetworkManager (pid 1527) is running...
```

The **service** command will report **NetworkManager is stopped** if the **NetworkManager** service is not running. To start it for the current session:

```
~]# service NetworkManager start
```

Run the **chkconfig** command to ensure that **NetworkManager** starts up every time the system boots:

```
~]# chkconfig NetworkManager on
```

For more information on starting, stopping and managing services and runlevels, see [Chapter 12, Services and Daemons](#).

10.2. INTERACTING WITH NETWORKMANAGER

Users do not interact with the **NetworkManager** system service directly. Instead, you can perform network configuration tasks via **NetworkManager**'s Notification Area applet. The applet has multiple states that serve as visual indicators for the type of connection you are currently using. Hover the pointer over the applet icon for tooltip information on the current connection state.



Figure 10.1. NetworkManager applet states

If you do not see the **NetworkManager** applet in the GNOME panel, and assuming that the **NetworkManager** package is installed on your system, you can start the applet by running the following command as a normal user (not **root**):

```
~]$ nm-applet &
```

After running this command, the applet appears in your Notification Area. You can ensure that the applet runs each time you log in by clicking **System → Preferences → Startup Applications** to open the **Startup Applications Preferences** window. Then, select the **Startup Programs** tab and check the box next to **NetworkManager**.

10.2.1. Connecting to a Network

When you left-click on the applet icon, you are presented with:

- a list of categorized networks you are currently connected to (such as **Wired** and **Wireless**);
- a list of all **Available Networks** that **NetworkManager** has detected;
- options for connecting to any configured Virtual Private Networks (VPNs); and,
- options for connecting to hidden or new wireless networks.

If you are connected to a network, its name is presented in bold typeface under its network type, such as **Wired** or **Wireless**. When many networks are available, such as wireless access points, the **More networks** expandable menu entry appears.



Figure 10.2. The NetworkManager applet's left-click menu, showing all available and connected-to networks

10.2.2. Configuring New and Editing Existing Connections

Next, right-click on the NetworkManager applet to open its context menu, which is the main point of entry for interacting with NetworkManager to configure connections.

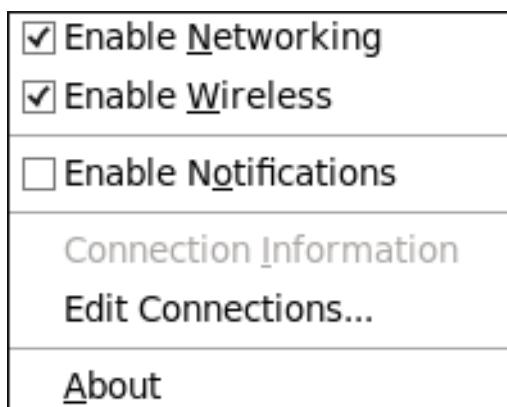


Figure 10.3. The NetworkManager applet's context menu

Ensure that the **Enable Networking** box is checked. If the system has detected a wireless card, then you will also see an **Enable Wireless** menu option. Check the **Enable Wireless** check box as well.

NetworkManager notifies you of network connection status changes if you check the **Enable Notifications** box. Clicking the **Connection Information** entry presents an informative **Connection Information** window that lists the connection type and interface, your IP address and routing details, and so on.

Finally, clicking on **Edit Connections** opens the **Network Connections** window, from where you can perform most of your network configuration tasks. Note that this window can also be opened by running, as a normal user:

```
~]$ nm-connection-editor &
```

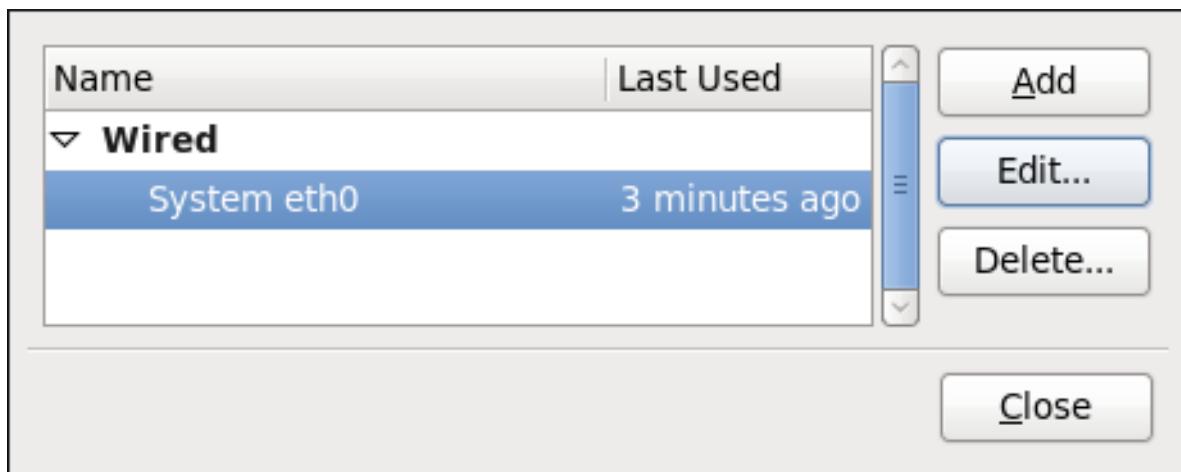


Figure 10.4. Configure networks using the Network Connections window

There is an arrow head symbol to the left which can be clicked to hide and reveal entries as needed. To create a new connection, click the **Add** button to view the selection list, select the connection type and click the **Create** button. Alternatively, to edit an existing connection select the interface name from the list and click the **Edit** button.

Then, to configure:

- wired Ethernet connections, proceed to [Section 10.3.1, "Establishing a Wired \(Ethernet\) Connection"](#);
- wireless connections, proceed to [Section 10.3.2, "Establishing a Wireless Connection"](#); or,
- mobile broadband connections, proceed to [Section 10.3.3, "Establishing a Mobile Broadband Connection"](#); or,
- VPN connections, proceed to [Section 10.3.4, "Establishing a VPN Connection"](#).

10.2.3. Connecting to a Network Automatically

For any connection type you add or configure, you can choose whether you want **NetworkManager** to try to connect to that network automatically when it is available.

Procedure 10.1. Configuring NetworkManager to Connect to a Network Automatically When Detected

1. Right-click on the **NetworkManager** applet icon in the Notification Area and click **Edit Connections**. The **Network Connections** window appears.
2. Click the arrow head if necessary to reveal the list of connections.
3. Select the specific connection that you want to configure and click **Edit**.
4. Check **Connect automatically** to cause **NetworkManager** to auto-connect to the connection whenever **NetworkManager** detects that it is available. Uncheck the check box if you do not want **NetworkManager** to connect automatically. If the box is unchecked, you will have to select that connection manually in the **NetworkManager** applet's left-click menu to cause it to connect.

10.2.4. User and System Connections

NetworkManager connections are always either *user connections* or *system connections*. Depending on the system-specific policy that the administrator has configured, users may need **root** privileges to create and modify system connections. **NetworkManager**'s default policy enables users to create and modify user connections, but requires them to have **root** privileges to add, modify or delete system connections.

User connections are so-called because they are specific to the user who creates them. In contrast to system connections, whose configurations are stored under the **/etc/sysconfig/network-scripts/** directory (mainly in **ifcfg-<network_type>** interface configuration files), user connection settings are stored in the GConf configuration database and the GNOME keyring, and are only available during login sessions for the user who created them. Thus, logging out of the desktop session causes user-specific connections to become unavailable.



NOTE

Because **NetworkManager** uses the GConf and GNOME keyring applications to store user connection settings, and because these settings are specific to your desktop session, it is highly recommended to configure your personal VPN connections as user connections. If you do so, other Non-**root** users on the system cannot view or access these connections in any way.

System connections, on the other hand, become available at boot time and can be used by other users on the system without first logging in to a desktop session.

NetworkManager can quickly and conveniently convert user to system connections and vice versa. Converting a user connection to a system connection causes **NetworkManager** to create the relevant interface configuration files under the **/etc/sysconfig/network-scripts/** directory, and to delete the GConf settings from the user's session. Conversely, converting a system to a user-specific connection causes **NetworkManager** to remove the system-wide configuration files and create the corresponding GConf/GNOME keyring settings.

Connection name:	System eth0
<input checked="" type="checkbox"/> Connect automatically <input checked="" type="checkbox"/> Available to all users	

Figure 10.5. The **Available to all users** check box controls whether connections are user-specific or system-wide

Procedure 10.2. Changing a Connection to be User-Specific instead of System-Wide, or Vice-Versa



NOTE

Depending on the system's policy, you may need **root** privileges on the system in order to change whether a connection is user-specific or system-wide.

1. Right-click on the **NetworkManager** applet icon in the Notification Area and click **Edit Connections**. The **Network Connections** window appears.

2. If needed, select the arrow head (on the left hand side) to hide and reveal the types of available network connections.
3. Select the specific connection that you want to configure and click **Edit**.
4. Check the **Available to all users** check box to ask **NetworkManager** to make the connection a system-wide connection. Depending on system policy, you may then be prompted for the **root** password by the **PolicyKit** application. If so, enter the **root** password to finalize the change.

Conversely, uncheck the **Available to all users** check box to make the connection user-specific.

10.3. ESTABLISHING CONNECTIONS

10.3.1. Establishing a Wired (Ethernet) Connection

To establish a wired network connection, Right-click on the **NetworkManager** applet to open its context menu, ensure that the **Enable Networking** box is checked, then click on **Edit Connections**. This opens the **Network Connections** window. Note that this window can also be opened by running, as a normal user:

```
~]$ nm-connection-editor &
```

You can click on the arrow head to reveal and hide the list of connections as needed.

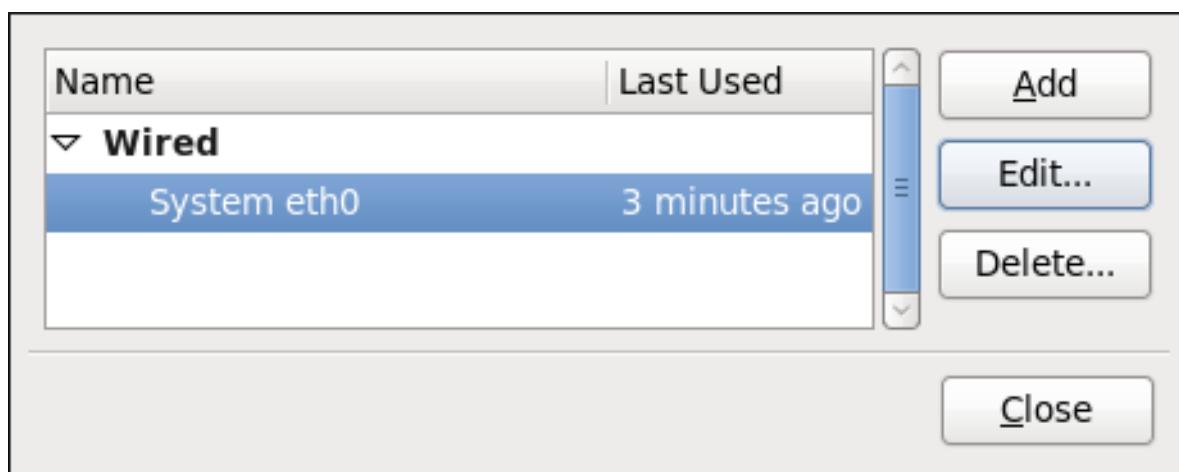


Figure 10.6. The Network Connections window showing the newly created System eth0 connection

The system startup scripts create and configure a single wired connection called **System eth0** by default on all systems. Although you can edit **System eth0**, creating a new wired connection for your custom settings is recommended. You can create a new wired connection by clicking the **Add** button, selecting the **Wired** entry from the list that appears and then clicking the **Create** button.

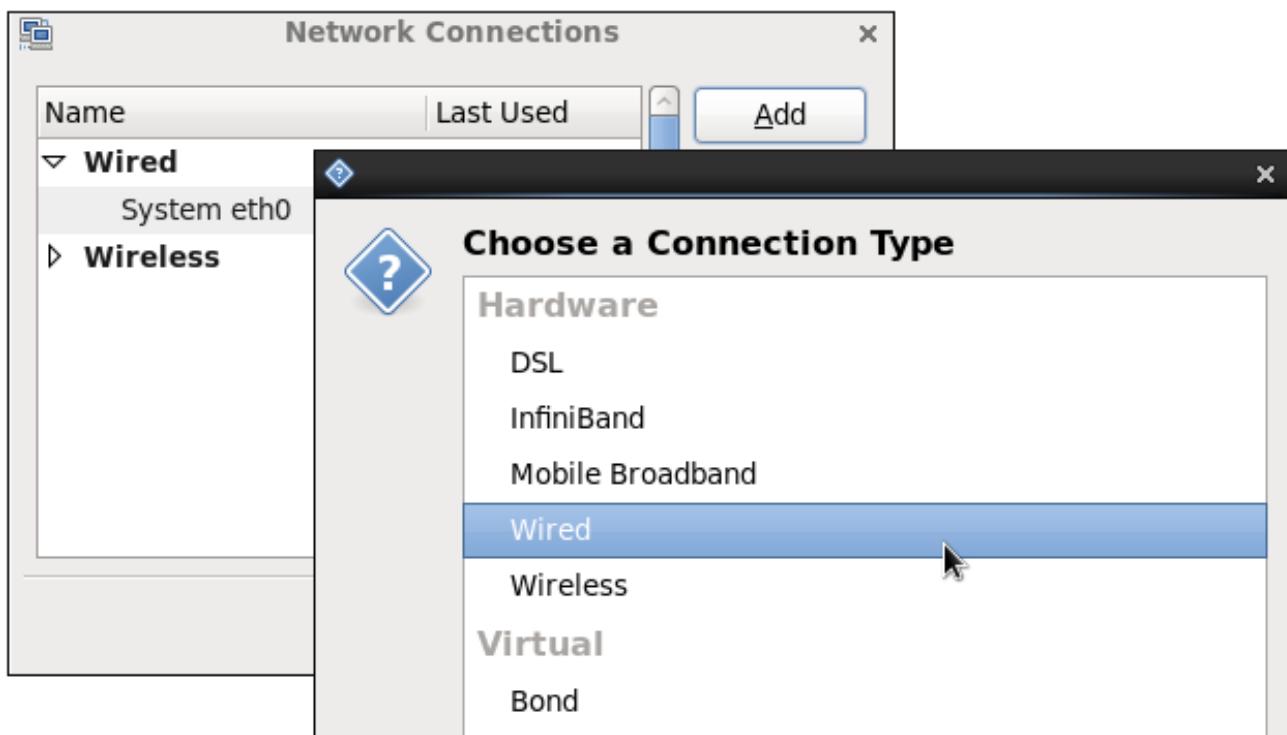


Figure 10.7. Selecting a new connection type from the "Choose a Connection Type" list



NOTE

When you add a new connection by clicking the **Add** button, a list of connection types appears. Once you have made a selection and clicked on the **Create** button, **NetworkManager** creates a new configuration file for that connection and then opens the same dialog that is used for editing an existing connection. There is no difference between these dialogs. In effect, you are always editing a connection; the difference only lies in whether that connection previously existed or was just created by **NetworkManager** when you clicked **Create**.

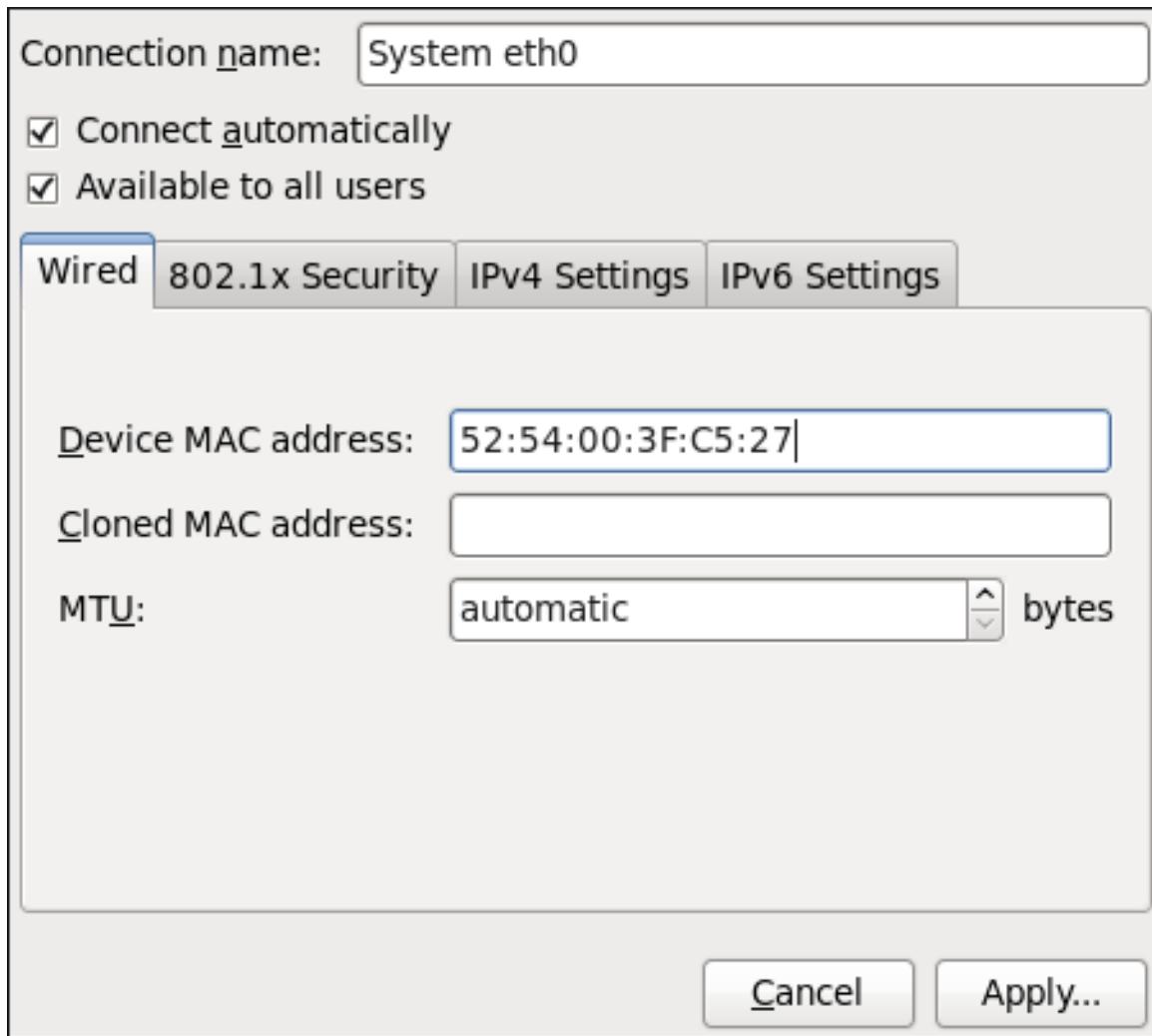


Figure 10.8. Editing the newly created Wired connection System eth0

Configuring the Connection Name, Auto-Connect Behavior, and Availability Settings

Three settings in the **Editing** dialog are common to all connection types:

- **Connection name** – Enter a descriptive name for your network connection. This name will be used to list this connection in the **Wired** section of the **Network Connections** window.
- **Connect automatically** – Check this box if you want **NetworkManager** to auto-connect to this connection when it is available. See [Section 10.2.3, “Connecting to a Network Automatically”](#) for more information.
- **Available to all users** – Check this box to create a connection available to all users on the system. Changing this setting may require **root** privileges. See [Section 10.2.4, “User and System Connections”](#) for details.

Configuring the Wired Tab

The final three configurable settings are located within the **Wired** tab itself: the first is a text-entry field where you can specify a MAC (Media Access Control) address, and the second allows you to specify a cloned MAC address, and third allows you to specify the MTU (Maximum Transmission Unit) value. Normally, you can leave the **MAC address** field blank and the **MTU** set to **automatic**. These defaults will suffice unless you are associating a wired connection with a second or specific NIC, or performing advanced networking. In such cases, see the following descriptions:

MAC Address

Network hardware such as a Network Interface Card (NIC) has a unique MAC address (Media Access

Control; also known as a *hardware address*) that identifies it to the system. Running the **ip addr** command will show the MAC address associated with each interface. For example, in the following **ip addr** output, the MAC address for the **eth0** interface (which is **52:54:00:26:9e:f1**) immediately follows the **link/ether** keyword:

```
~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN qlen 1000
    link/ether 52:54:00:26:9e:f1 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.251/24 brd 192.168.122.255 scope global eth0
        inet6 fe80::5054:ff:fe26:9ef1/64 scope link
            valid_lft forever preferred_lft forever
```

A single system can have one or more NICs installed on it. The **MAC address** field therefore allows you to associate a specific NIC with a specific connection (or connections). As mentioned, you can determine the MAC address using the **ip addr** command, and then copy and paste that value into the **MAC address** text-entry field.

The cloned MAC address field is mostly for use in such situations where a network service has been restricted to a specific MAC address and you need to emulate that MAC address.

MTU

The MTU (Maximum Transmission Unit) value represents the size in bytes of the largest packet that the connection will use to transmit. This value defaults to **1500** when using IPv4, or a variable number **1280** or higher for IPv6, and does not generally need to be specified or changed.

Saving Your New (or Modified) Connection and Making Further Configurations

Once you have finished editing your wired connection, click the **Apply** button and **NetworkManager** will immediately save your customized configuration. Given a correct configuration, you can connect to your new or customized connection by selecting it from the **NetworkManager** Notification Area applet. See [Section 10.2.1, “Connecting to a Network”](#) for information on using your new or altered connection.

You can further configure an existing connection by selecting it in the **Network Connections** window and clicking **Edit** to return to the **Editing** dialog.

Then, to configure:

- port-based Network Access Control (PNAC), click the **802.1X Security** tab and proceed to [Section 10.3.9.1, “Configuring 802.1X Security”](#);
- IPv4 settings for the connection, click the **IPv4 Settings** tab and proceed to [Section 10.3.9.4, “Configuring IPv4 Settings”](#); or,
- IPv6 settings for the connection, click the **IPv6 Settings** tab and proceed to [Section 10.3.9.5, “Configuring IPv6 Settings”](#).

10.3.2. Establishing a Wireless Connection

This section explains how to use **NetworkManager** to configure a wireless (also known as Wi-Fi or 802.11a/b/g/n) connection to an Access Point.

To configure a mobile broadband (such as 3G) connection, see [Section 10.3.3, “Establishing a Mobile Broadband Connection”](#).

Quickly Connecting to an Available Access Point

The easiest way to connect to an available access point is to left-click on the **NetworkManager** applet, locate the *Service Set Identifier* (SSID) of the access point in the list of **Available** networks, and click on it. If the access point is secured, a dialog prompts you for authentication.

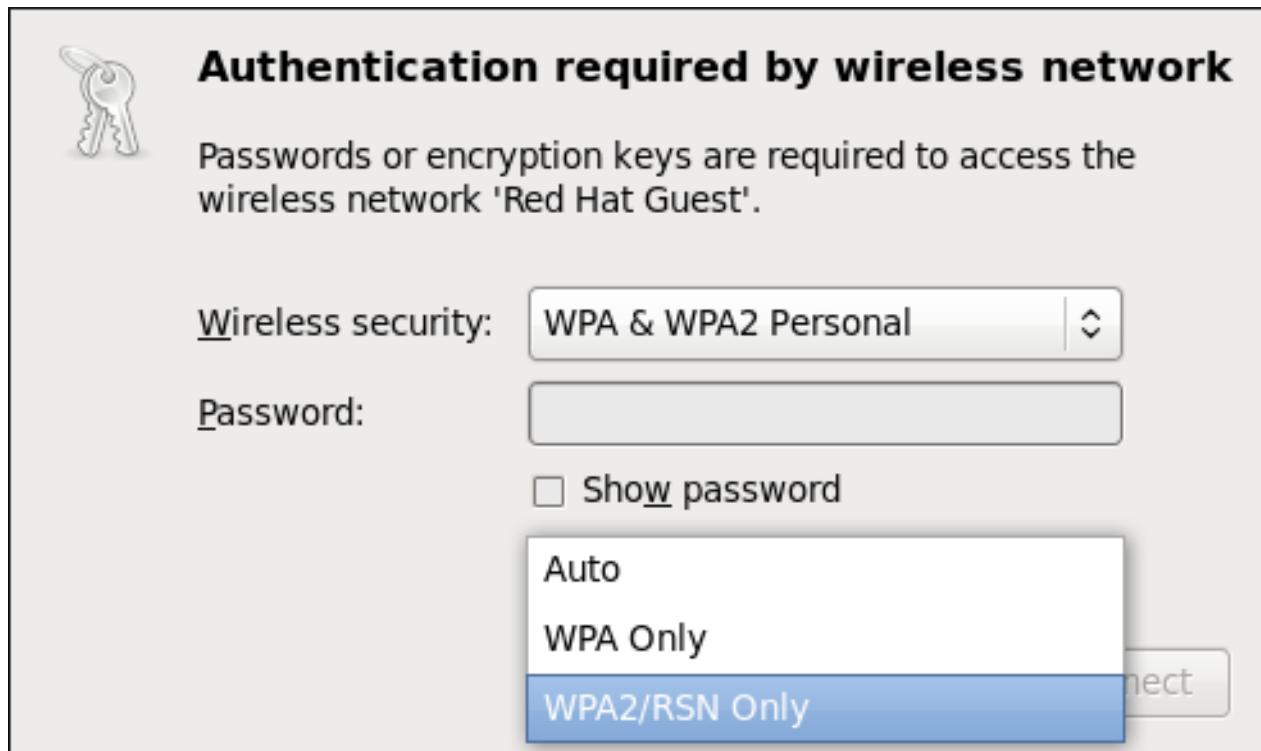


Figure 10.9. Authenticating to a wireless access point

NetworkManager tries to auto-detect the type of security used by the access point. If there are multiple possibilities, **NetworkManager** guesses the security type and presents it in the **Wireless security** dropdown menu. To see if there are multiple choices, click the **Wireless security** dropdown menu and select the type of security the access point is using. If you are unsure, try connecting to each type in turn. Finally, enter the key or passphrase in the **Password** field. Certain password types, such as a 40-bit WEP or 128-bit WPA key, are invalid unless they are of a requisite length. The **Connect** button will remain inactive until you enter a key of the length required for the selected security type. To learn more about wireless security, see [Section 10.3.9.2, “Configuring Wireless Security”](#).



NOTE

In the case of WPA and WPA2 (Personal and Enterprise), an option to select between Auto, WPA and WPA2 has been added. This option is intended for use with an access point that is offering both WPA and WPA2. Select one of the protocols if you would like to prevent roaming between the two protocols. Roaming between WPA and WPA2 on the same access point can cause loss of service.

If **NetworkManager** connects to the access point successfully, its applet icon will change into a graphical indicator of the wireless connection's signal strength.



Figure 10.10. Applet icon indicating a wireless connection signal strength of 75%

You can also edit the settings for one of these auto-created access point connections just as if you had added it yourself. The **Wireless** tab of the **Network Connections** window lists all of the connections you have ever tried to connect to: NetworkManager names each of them **Auto <SSID>**, where SSID is the Service Set identifier of the access point.

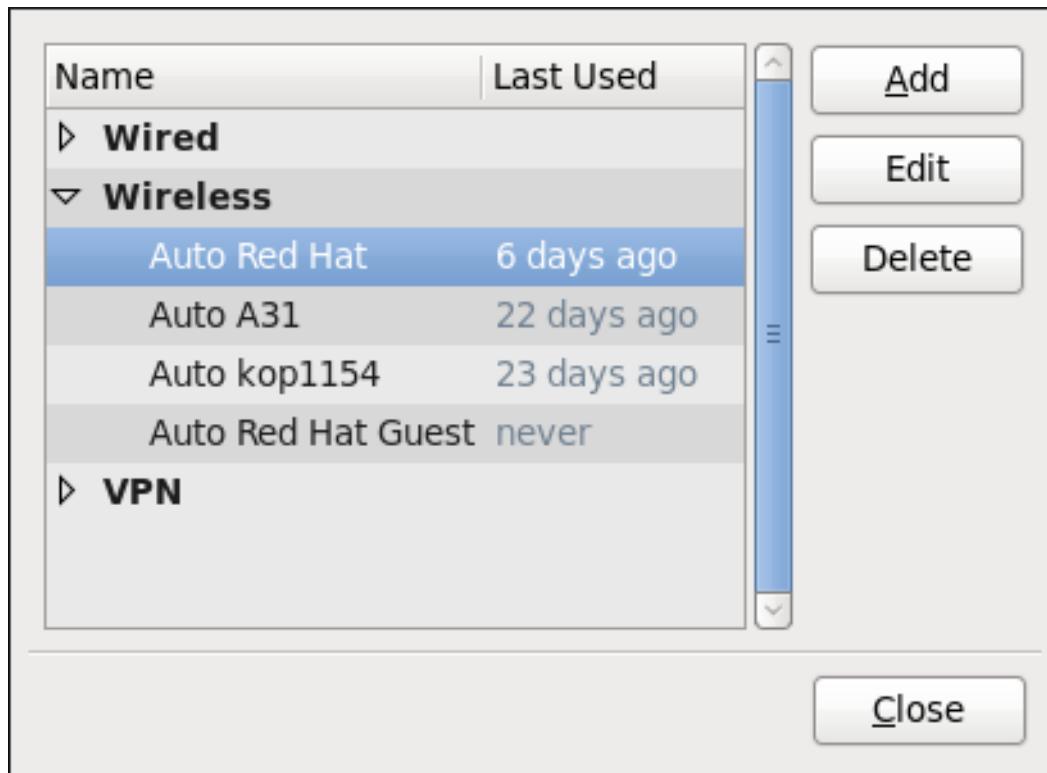


Figure 10.11. An example of access points that have previously been connected to

Connecting to a Hidden Wireless Network

All access points have a *Service Set Identifier* (SSID) to identify them. However, an access point may be configured not to broadcast its SSID, in which case it is *hidden*, and will not show up in NetworkManager's list of **Available** networks. You can still connect to a wireless access point that is hiding its SSID as long as you know its SSID, authentication method, and secrets.

To connect to a hidden wireless network, left-click NetworkManager's applet icon and select **Connect to Hidden Wireless Network** to cause a dialog to appear. If you have connected to the hidden network before, use the **Connection** dropdown to select it, and click **Connect**. If you have not, leave the **Connection** dropdown as **New**, enter the SSID of the hidden network, select its **Wireless security** method, enter the correct authentication secrets, and click **Connect**.

For more information on wireless security settings, see [Section 10.3.9.2, “Configuring Wireless Security”](#).

Editing a Connection, or Creating a Completely New One

You can edit an existing connection that you have tried or succeeded in connecting to in the past by opening the **Wireless** tab of the **Network Connections**, selecting the connection by name (words which follow **Auto** refer to the SSID of an access point), and clicking **Edit**.

You can create a new connection by opening the **Network Connections** window, clicking the **Add** button, selecting **Wireless**, and clicking the **Create** button.

1. Right-click on the **NetworkManager** applet icon in the Notification Area and click **Edit Connections**. The **Network Connections** window appears.
2. Click the **Add** button.
3. Select the **Wireless** entry from the list.
4. Click the **Create** button.



Figure 10.12. Editing the newly created Wireless connection 1

Configuring the Connection Name, Auto-Connect Behavior, and Availability Settings
Three settings in the **Editing** dialog are common to all connection types:

- **Connection name** – Enter a descriptive name for your network connection. This name will be used to list this connection in the **Wireless** section of the **Network Connections** window. By default, wireless connections are named the same as the **SSID** of the wireless access point. You can rename the wireless connection without affecting its ability to connect, but it is recommended to retain the **SSID** name.

- **Connect automatically** – Check this box if you want **NetworkManager** to auto-connect to this connection when it is available. See [Section 10.2.3, “Connecting to a Network Automatically”](#) for more information.
- **Available to all users** – Check this box to create a connection available to all users on the system. Changing this setting may require **root** privileges. See [Section 10.2.4, “User and System Connections”](#) for details.

Configuring the Wireless Tab

SSID

All access points have a *Service Set identifier* to identify them. However, an access point may be configured not to broadcast its SSID, in which case it is *hidden*, and will not show up in **NetworkManager**'s list of **Available** networks. You can still connect to a wireless access point that is hiding its SSID as long as you know its SSID (and authentication secrets).

For information on connecting to a hidden wireless network, see [the section called “Connecting to a Hidden Wireless Network”](#).

Mode

Infrastructure – Set **Mode** to **Infrastructure** if you are connecting to a dedicated wireless access point or one built into a network device such as a router or a switch.

Ad-hoc – Set **Mode** to **Ad-hoc** if you are creating a peer-to-peer network for two or more mobile devices to communicate directly with each other. If you use **Ad-hoc** mode, referred to as *Independent Basic Service Set* (IBSS) in the 802.11 standard, you must ensure that the same **SSID** is set for all participating wireless devices, and that they are all communicating over the same channel.

BSSID

The Basic Service Set Identifier (BSSID) is the MAC address of the specific wireless access point you are connecting to when in **Infrastructure** mode. This field is blank by default, and you are able to connect to a wireless access point by **SSID** without having to specify its **BSSID**. If the BSSID is specified, it will force the system to associate to a specific access point only.

For ad-hoc networks, the **BSSID** is generated randomly by the **mac80211** subsystem when the ad-hoc network is created. It is not displayed by **NetworkManager**

MAC address

Like an Ethernet Network Interface Card (NIC), a wireless adapter has a unique MAC address (Media Access Control; also known as a *hardware address*) that identifies it to the system. Running the **ip addr** command will show the MAC address associated with each interface. For example, in the following **ip addr** output, the MAC address for the **wlan0** interface (which is **00:1c:bf:02:f8:70**) immediately follows the **link/ether** keyword:

```
~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN qlen 1000
    link/ether 52:54:00:26:9e:f1 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.251/24 brd 192.168.122.255 scope global eth0
```

```

inet6 fe80::5054:ff:fe26:9ef1/64 scope link
    valid_lft forever preferred_lft forever
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
    link/ether 00:1c:bf:02:f8:70 brd ff:ff:ff:ff:ff:ff
    inet 10.200.130.67/24 brd 10.200.130.255 scope global wlan0
        inet6 fe80::21c:bfff:fe02:f870/64 scope link
            valid_lft forever preferred_lft forever

```

A single system could have one or more wireless network adapters connected to it. The **MAC address** field therefore allows you to associate a specific wireless adapter with a specific connection (or connections). As mentioned, you can determine the MAC address using the **ip addr** command, and then copy and paste that value into the **MAC address** text-entry field.

MTU

The MTU (Maximum Transmission Unit) value represents the size in bytes of the largest packet that the connection will use to transmit. If set to a non-zero number, only packets of the specified size or smaller will be transmitted. Larger packets are broken up into multiple Ethernet frames. It is recommended to leave this setting on **automatic**.

Saving Your New (or Modified) Connection and Making Further Configurations

Once you have finished editing the wireless connection, click the **Apply** button and **NetworkManager** will immediately save your customized configuration. Given a correct configuration, you can successfully connect to your the modified connection by selecting it from the **NetworkManager** Notification Area applet. See [Section 10.2.1, “Connecting to a Network”](#) for details on selecting and connecting to a network.

You can further configure an existing connection by selecting it in the **Network Connections** window and clicking **Edit** to return to the **Editing** dialog.

Then, to configure:

- security authentication for the wireless connection, click the **Wireless Security** tab and proceed to [Section 10.3.9.2, “Configuring Wireless Security”](#);
- IPv4 settings for the connection, click the **IPv4 Settings** tab and proceed to [Section 10.3.9.4, “Configuring IPv4 Settings”](#); or,
- IPv6 settings for the connection, click the **IPv6 Settings** tab and proceed to [Section 10.3.9.5, “Configuring IPv6 Settings”](#).

10.3.3. Establishing a Mobile Broadband Connection

You can use **NetworkManager**'s mobile broadband connection abilities to connect to the following 2G and 3G services:

- 2G – GPRS (*General Packet Radio Service*) or EDGE (*Enhanced Data Rates for GSM Evolution*)
- 3G – UMTS (*Universal Mobile Telecommunications System*) or HSPA (*High Speed Packet Access*)

Your computer must have a mobile broadband device (modem), which the system has discovered and recognized, in order to create the connection. Such a device may be built into your computer (as is the case on many notebooks and netbooks), or may be provided separately as internal or external hardware. Examples include PC card, USB Modem or Dongle, mobile or cellular telephone capable of acting as a modem.

Procedure 10.3. Adding a New Mobile Broadband Connection

You can configure a mobile broadband connection by opening the **Network Connections** window, clicking **Add**, and selecting **Mobile Broadband** from the list.

1. Right-click on the **NetworkManager** applet icon in the Notification Area and click **Edit Connections**. The **Network Connections** window appears.
2. Click the **Add** button to open the selection list. Select **Mobile Broadband** and then click **Create**. The **Set up a Mobile Broadband Connection** assistant appears.
3. Under **Create a connection for this mobile broadband device**, choose the 2G- or 3G-capable device you want to use with the connection. If the dropdown menu is inactive, this indicates that the system was unable to detect a device capable of mobile broadband. In this case, click **Cancel**, ensure that you do have a mobile broadband-capable device attached and recognized by the computer and then retry this procedure. Click the **Forward** button.
4. Select the country where your service provider is located from the list and click the **Forward** button.
5. Select your provider from the list or enter it manually. Click the **Forward** button.
6. Select your payment plan from the dropdown menu and confirm the *Access Point Name* (APN) is correct. Click the **Forward** button.
7. Review and confirm the settings and then click the **Apply** button.
8. Edit the mobile broadband-specific settings by referring to the *Configuring the Mobile Broadband Tab* description below .

Procedure 10.4. Editing an Existing Mobile Broadband Connection

Follow these steps to edit an existing mobile broadband connection.

1. Right-click on the **NetworkManager** applet icon in the Notification Area and click **Edit Connections**. The **Network Connections** window appears.
2. Select the connection you want to edit and click the **Edit** button.
3. Select the **Mobile Broadband** tab.
4. Configure the connection name, auto-connect behavior, and availability settings.

Three settings in the **Editing** dialog are common to all connection types:

- **Connection name** – Enter a descriptive name for your network connection. This name will be used to list this connection in the **Mobile Broadband** section of the **Network Connections** window.
- **Connect automatically** – Check this box if you want **NetworkManager** to auto-connect to this connection when it is available. See [Section 10.2.3, "Connecting to a Network Automatically"](#) for more information.
- **Available to all users** – Check this box to create a connection available to all users on the system. Changing this setting may require **root** privileges. See [Section 10.2.4, "User and System Connections"](#) for details.

5. Edit the mobile broadband-specific settings by referring to the *Configuring the Mobile Broadband Tab* description below .

Saving Your New (or Modified) Connection and Making Further Configurations

Once you have finished editing your mobile broadband connection, click the **Apply** button and **NetworkManager** will immediately save your customized configuration. Given a correct configuration, you can connect to your new or customized connection by selecting it from the **NetworkManager** Notification Area applet. See [Section 10.2.1, “Connecting to a Network”](#) for information on using your new or altered connection.

You can further configure an existing connection by selecting it in the **Network Connections** window and clicking **Edit** to return to the **Editing** dialog.

Then, to configure:

- Point-to-point settings for the connection, click the **PPP Settings** tab and proceed to [Section 10.3.9.3, “Configuring PPP \(Point-to-Point\) Settings”](#);
- IPv4 settings for the connection, click the **IPv4 Settings** tab and proceed to [Section 10.3.9.4, “Configuring IPv4 Settings”](#); or,
- IPv6 settings for the connection, click the **IPv6 Settings** tab and proceed to [Section 10.3.9.5, “Configuring IPv6 Settings”](#).

Configuring the Mobile Broadband Tab

If you have already added a new mobile broadband connection using the assistant (see [Procedure 10.3, “Adding a New Mobile Broadband Connection”](#) for instructions), you can edit the **Mobile Broadband** tab to disable roaming if home network is not available, assign a network ID, or instruct **NetworkManager** to prefer a certain technology (such as 3G or 2G) when using the connection.

Number

The number that is dialed to establish a PPP connection with the GSM-based mobile broadband network. This field may be automatically populated during the initial installation of the broadband device. You can usually leave this field blank and enter the **APN** instead.

Username

Enter the user name used to authenticate with the network. Some providers do not provide a user name, or accept any user name when connecting to the network.

Password

Enter the password used to authenticate with the network. Some providers do not provide a password, or accept any password.

APN

Enter the Access Point Name (APN) used to establish a connection with the GSM-based network. Entering the correct APN for a connection is important because it often determines:

- how the user is billed for their network usage; and/or
- whether the user has access to the Internet, an intranet, or a subnetwork.

Network ID

Entering a **Network ID** causes **NetworkManager** to force the device to register only to a specific network. This can be used to ensure the connection does not roam when it is not possible to control roaming directly.

Type

Any – The default value of **Any** leaves the modem to select the fastest network.

3G (UMTS/HSPA) – Force the connection to use only 3G network technologies.

2G (GPRS/EDGE) – Force the connection to use only 2G network technologies.

Prefer 3G (UMTS/HSPA) – First attempt to connect using a 3G technology such as HSPA or UMTS, and fall back to GPRS or EDGE only upon failure.

Prefer 2G (GPRS/EDGE) – First attempt to connect using a 2G technology such as GPRS or EDGE, and fall back to HSPA or UMTS only upon failure.

Allow roaming if home network is not available

Uncheck this box if you want **NetworkManager** to terminate the connection rather than transition from the home network to a roaming one, thereby avoiding possible roaming charges. If the box is checked, **NetworkManager** will attempt to maintain a good connection by transitioning from the home network to a roaming one, and vice versa.

PIN

If your device's *SIM* (*Subscriber Identity Module*) is locked with a *PIN* (*Personal Identification Number*), enter the PIN so that **NetworkManager** can unlock the device. **NetworkManager** must unlock the SIM if a PIN is required in order to use the device for any purpose.

10.3.4. Establishing a VPN Connection

Establishing an encrypted Virtual Private Network (VPN) enables you to communicate securely between your Local Area Network (LAN), and another, remote LAN. After successfully establishing a VPN connection, a VPN router or gateway performs the following actions upon the packets you transmit:

1. it adds an *Authentication Header* for routing and authentication purposes;
2. it encrypts the packet data; and,
3. it encloses the data with an Encapsulating Security Payload (ESP), which constitutes the decryption and handling instructions.

The receiving VPN router strips the header information, decrypts the data, and routes it to its intended destination (either a workstation or other node on a network). Using a network-to-network connection, the receiving node on the local network receives the packets already decrypted and ready for processing. The encryption/decryption process in a network-to-network VPN connection is therefore transparent to clients.

Because they employ several layers of authentication and encryption, VPNs are a secure and effective means of connecting multiple remote nodes to act as a unified intranet.

Procedure 10.5. Adding a New VPN Connection

1. You can configure a new VPN connection by opening the **Network Connections** window, clicking the **Add** button and selecting a type of VPN from the **VPN** section of the new connection list.
2. Right-click on the **NetworkManager** applet icon in the Notification Area and click **Edit Connections**. The **Network Connections** window appears.
3. Click the **Add** button.
4. The **Choose a Connection Type** list appears.

**NOTE**

The appropriate **NetworkManager** VPN plug-in for the VPN type you want to configure must be installed (see [Section 8.2.4, “Installing Packages”](#) for more information on how to install new packages in Red Hat Enterprise Linux 6).

The **VPN** section in the **Choose a Connection Type** list will **not** appear if you do not have a suitable plug-in installed.

5. Select the VPN protocol for the gateway you are connecting to from the **Choose a Connection Type** list. The VPN protocols available for selection in the list correspond to the **NetworkManager** VPN plug-ins installed. For example, if `NetworkManager-openswan`, the **NetworkManager** VPN plug-in for `libreswan`, is installed, then the IPsec based VPN will be selectable from the **Choose a Connection Type** list.

**NOTE**

In Red Hat Enterprise Linux 6.8, `openswan` has been obsoleted by `libreswan`. `NetworkManager-openswan` has been modified to support both `openswan` and `libreswan`.

After selecting the correct one, press the **Create** button.

7. The **Editing VPN Connection 1** window then appears. This window presents settings customized for the type of VPN connection you selected in [Step 6](#).

Procedure 10.6. Editing an Existing VPN Connection

You can configure an existing VPN connection by opening the **Network Connections** window and selecting the name of the connection from the list. Then click the **Edit** button.

1. Right-click on the **NetworkManager** applet icon in the Notification Area and click **Edit Connections**. The **Network Connections** window appears.
2. Select the connection you want to edit and click the **Edit** button.

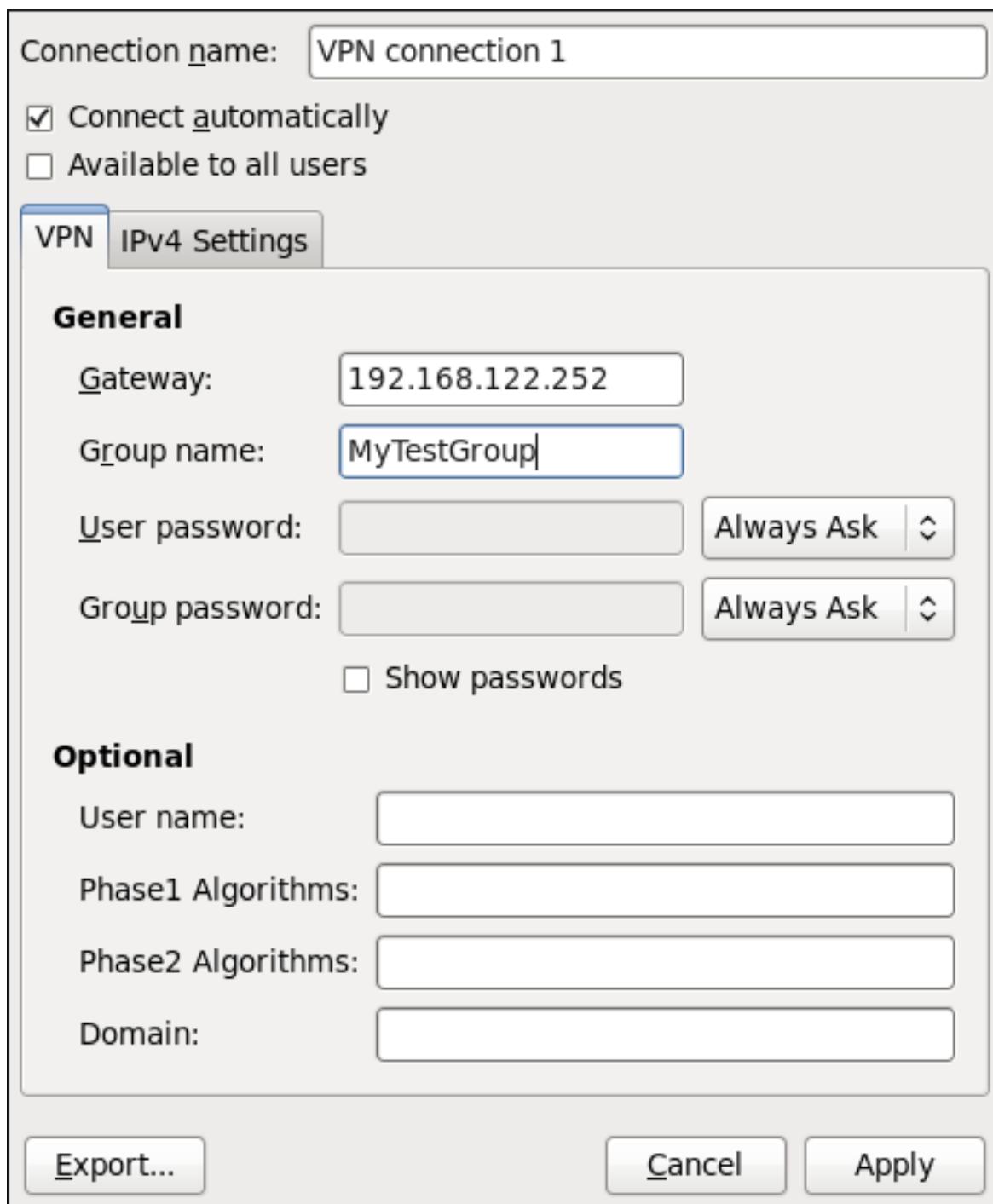


Figure 10.13. Editing the newly created IPsec VPN connection 1

Configuring the Connection Name, Auto-Connect Behavior, and Availability Settings
 Three settings in the **Editing** dialog are common to all connection types:

- **Connection name** – Enter a descriptive name for your network connection. This name will be used to list this connection in the **VPN** section of the **Network Connections** window.
- **Connect automatically** – Check this box if you want **NetworkManager** to auto-connect to this connection when it is available. See [Section 10.2.3, “Connecting to a Network Automatically”](#) for more information.
- **Available to all users** – Check this box to create a connection available to all users on the system. Changing this setting may require **root** privileges. See [Section 10.2.4, “User and System Connections”](#) for details.

Configuring the VPN Tab

Gateway

The name or IP address of the remote VPN gateway.

Group name

The name of a VPN group configured on the remote gateway.

User password

If required, enter the password used to authenticate with the VPN.

Group password

If required, enter the password used to authenticate with the VPN.

User name

If required, enter the user name used to authenticate with the VPN.

Phase1 Algorithms

If required, enter the algorithms to be used to authenticate and set up an encrypted channel.

Phase2 Algorithms

If required, enter the algorithms to be used for the IPsec negotiations.

Domain

If required, enter the Domain Name.

NAT traversal

Cisco UDP (default) – IPsec over UDP.

NAT-T – ESP encapsulation and IKE extensions are used to handle NAT Traversal.

Disabled – No special NAT measures required.

Disable Dead Peer Detection – Disable the sending of probes to the remote gateway or endpoint.

Saving Your New (or Modified) Connection and Making Further Configurations

Once you have finished editing your new VPN connection, click the **Apply** button and **NetworkManager** will immediately save your customized configuration. Given a correct configuration, you can connect to your new or customized connection by selecting it from the **NetworkManager** Notification Area applet. See [Section 10.2.1, “Connecting to a Network”](#) for information on using your new or altered connection.

You can further configure an existing connection by selecting it in the **Network Connections** window and clicking **Edit** to return to the **Editing** dialog.

Then, to configure:

- IPv4 settings for the connection, click the **IPv4 Settings** tab and proceed to [Section 10.3.9.4, “Configuring IPv4 Settings”](#).

10.3.5. Establishing a DSL Connection

This section is intended for those installations which have a DSL card fitted within a host rather than the external combined DSL modem router combinations typical of private consumer or SOHO installations.

Procedure 10.7. Adding a New DSL Connection

You can configure a new DSL connection by opening the **Network Connections** window, clicking the **Add** button and selecting **DSL** from the **Hardware** section of the new connection list.

1. Right-click on the **NetworkManager** applet icon in the Notification Area and click **Edit Connections**. The **Network Connections** window appears.
2. Click the **Add** button.
3. The **Choose a Connection Type** list appears.
4. Select **DSL** and press the **Create** button.
5. The **Editing DSL Connection 1** window appears.

Procedure 10.8. Editing an Existing DSL Connection

You can configure an existing DSL connection by opening the **Network Connections** window and selecting the name of the connection from the list. Then click the **Edit** button.

1. Right-click on the **NetworkManager** applet icon in the Notification Area and click **Edit Connections**. The **Network Connections** window appears.
2. Select the connection you want to edit and click the **Edit** button.

Configuring the Connection Name, Auto-Connect Behavior, and Availability Settings

Three settings in the **Editing** dialog are common to all connection types:

- **Connection name** – Enter a descriptive name for your network connection. This name will be used to list this connection in the **DSL** section of the **Network Connections** window.
- **Connect automatically** – Check this box if you want **NetworkManager** to auto-connect to this connection when it is available. See [Section 10.2.3, “Connecting to a Network Automatically”](#) for more information.
- **Available to all users** – Check this box to create a connection available to all users on the system. Changing this setting may require **root** privileges. See [Section 10.2.4, “User and System Connections”](#) for details.

Configuring the DSL Tab

Username

Enter the user name used to authenticate with the service provider.

Service

Leave blank unless otherwise directed.

Password

Enter the password supplied by the service provider.

Saving Your New (or Modified) Connection and Making Further Configurations

Once you have finished editing your DSL connection, click the **Apply** button and **NetworkManager** will immediately save your customized configuration. Given a correct configuration, you can connect to your new or customized connection by selecting it from the **NetworkManager** Notification Area applet. See [Section 10.2.1, "Connecting to a Network"](#) for information on using your new or altered connection.

You can further configure an existing connection by selecting it in the **Network Connections** window and clicking **Edit** to return to the **Editing** dialog.

Then, to configure:

- The MAC address and MTU settings, click the **Wired** tab and proceed to [the section called "Configuring the Wired Tab"](#);
- Point-to-point settings for the connection, click the **PPP Settings** tab and proceed to [Section 10.3.9.3, "Configuring PPP \(Point-to-Point\) Settings"](#);
- IPv4 settings for the connection, click the **IPv4 Settings** tab and proceed to [Section 10.3.9.4, "Configuring IPv4 Settings"](#).

10.3.6. Establishing a Bond Connection

You can use **NetworkManager** to create a Bond from two or more Wired or Infiniband connections. It is not necessary to create the connections to be bonded first. They can be configured as part of the process to configure the bond. You must have the MAC addresses of the interfaces available in order to complete the configuration process.



NOTE

NetworkManager support for bonding must be enabled by means of the **NM_BOND_VLAN_ENABLED** directive and then **NetworkManager** must be restarted. See [Section 11.2.1, "Ethernet Interfaces"](#) for an explanation of **NM_CONTROLLED** and the **NM_BOND_VLAN_ENABLED** directive. See [Section 12.3.4, "Restarting a Service"](#) for an explanation of restarting a service such as **NetworkManager** from the command line. Alternatively, for a graphical tool see [Section 12.2.1, "Using the Service Configuration Utility"](#).

Procedure 10.9. Adding a New Bond Connection

You can configure a Bond connection by opening the **Network Connections** window, clicking **Add**, and selecting **Bond** from the list.

1. Right-click on the **NetworkManager** applet icon in the Notification Area and click **Edit Connections**. The **Network Connections** window appears.
2. Click the **Add** button to open the selection list. Select **Bond** and then click **Create**. The **Editing Bond connection 1** window appears.
3. On the **Bond** tab, click **Add** and select the type of interface you want to use with the bond connection. Click the **Create** button. Note that the dialog to select the slave type only comes up when you create the first slave; after that, it will automatically use that same type for all further slaves.
4. The **Editing bond0 slave 1** window appears. Fill in the MAC address of the first interface to be bonded. The first slave's MAC address will be used as the MAC address for the bond interface. If required, enter a clone MAC address to be used as the bond's MAC address. Click the **Apply**

button.

5. The **Authenticate** window appears. Enter the **root** password to continue. Click the **Authenticate** button.
6. The name of the bonded slave appears in the **Bonded Connections window**. Click the **Add** button to add further slave connections.
7. Review and confirm the settings and then click the **Apply** button.
8. Edit the bond-specific settings by referring to [the section called "Configuring the Bond Tab" below](#).

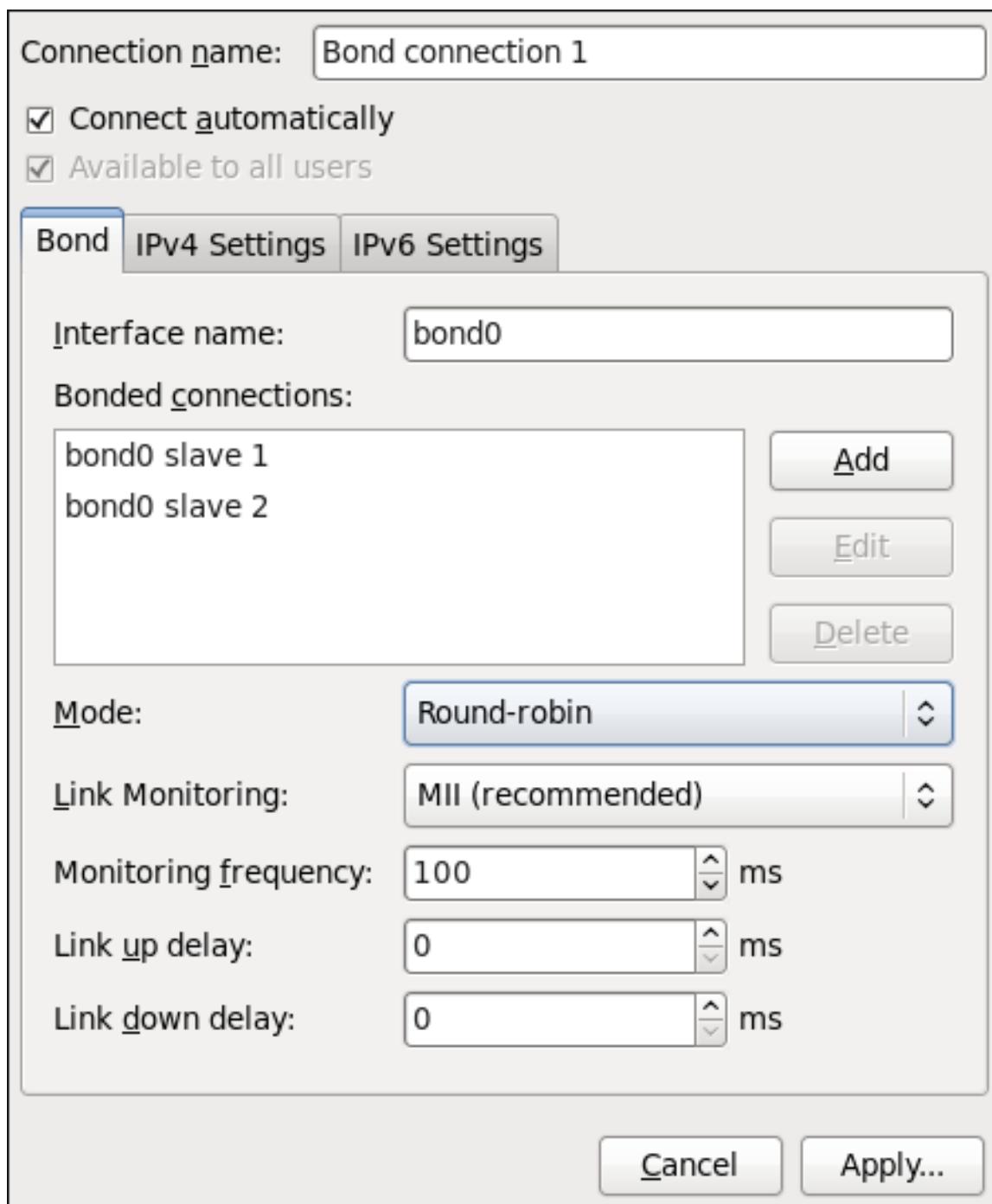


Figure 10.14. Editing the newly created Bond connection 1

Procedure 10.10. Editing an Existing Bond Connection

Follow these steps to edit an existing bond connection.

1. Right-click on the **NetworkManager** applet icon in the Notification Area and click **Edit Connections**. The **Network Connections** window appears.
2. Select the connection you want to edit and click the **Edit** button.
3. Select the **Bond** tab.
4. Configure the connection name, auto-connect behavior, and availability settings.

Three settings in the **Editing** dialog are common to all connection types:

- **Connection name** – Enter a descriptive name for your network connection. This name will be used to list this connection in the **Bond** section of the **Network Connections** window.
 - **Connect automatically** – Select this box if you want **NetworkManager** to auto-connect to this connection when it is available. See [Section 10.2.3, “Connecting to a Network Automatically”](#) for more information.
 - **Available to all users** – Select this box to create a connection available to all users on the system. Changing this setting may require **root** privileges. See [Section 10.2.4, “User and System Connections”](#) for details.
5. Edit the bond-specific settings by referring to [the section called “Configuring the Bond Tab”](#) below.

Saving Your New (or Modified) Connection and Making Further Configurations

Once you have finished editing your bond connection, click the **Apply** button to save your customized configuration. Given a correct configuration, you can connect to your new or customized connection by selecting it from the **NetworkManager** Notification Area applet. See [Section 10.2.1, “Connecting to a Network”](#) for information on using your new or altered connection.

You can further configure an existing connection by selecting it in the **Network Connections** window and clicking **Edit** to return to the **Editing** dialog.

Then, to configure:

- IPv4 settings for the connection, click the **IPv4 Settings** tab and proceed to [Section 10.3.9.4, “Configuring IPv4 Settings”](#); or,
- IPv6 settings for the connection, click the **IPv6 Settings** tab and proceed to [Section 10.3.9.5, “Configuring IPv6 Settings”](#).

Configuring the Bond Tab

If you have already added a new bond connection (see [Procedure 10.9, “Adding a New Bond Connection”](#) for instructions), you can edit the **Bond** tab to set the load sharing mode and the type of link monitoring to use to detect failures of a slave connection.

Mode

The mode that is used to share traffic over the slave connections which make up the bond. The default is **Round-robin**. Other load sharing modes, such as **802.3ad**, can be selected by means of the drop-down list.

Link Monitoring

The method of monitoring the slaves ability to carry network traffic.

The following modes of load sharing are selectable from the **Mode** drop-down list:

Round-robin

Sets a round-robin policy for fault tolerance and load balancing. Transmissions are received and sent out sequentially on each bonded slave interface beginning with the first one available. This mode might not work behind a bridge with virtual machines without additional switch configuration.

Active backup

Sets an active-backup policy for fault tolerance. Transmissions are received and sent out via the first available bonded slave interface. Another bonded slave interface is only used if the active bonded slave interface fails. Note that this is the only mode available for bonds of InfiniBand devices.

XOR

Sets an XOR (exclusive-or) policy. Transmissions are based on the selected hash policy. The default is to derive a hash by XOR of the source and destination MAC addresses multiplied by the modulo of the number of slave interfaces. In this mode traffic destined for specific peers will always be sent over the same interface. As the destination is determined by the MAC addresses this method works best for traffic to peers on the same link or local network. If traffic has to pass through a single router then this mode of traffic balancing will be suboptimal.

Broadcast

Sets a broadcast policy for fault tolerance. All transmissions are sent on all slave interfaces. This mode might not work behind a bridge with virtual machines without additional switch configuration.

802.3ad

Sets an IEEE **802.3ad** dynamic link aggregation policy. Creates aggregation groups that share the same speed and duplex settings. Transmits and receives on all slaves in the active aggregator. Requires a network switch that is **802.3ad** compliant.

Adaptive transmit load balancing

Sets an adaptive Transmit Load Balancing (TLB) policy for fault tolerance and load balancing. The outgoing traffic is distributed according to the current load on each slave interface. Incoming traffic is received by the current slave. If the receiving slave fails, another slave takes over the MAC address of the failed slave. This mode is only suitable for local addresses known to the kernel bonding module and therefore cannot be used behind a bridge with virtual machines.

Adaptive load balancing

Sets an Adaptive Load Balancing (ALB) policy for fault tolerance and load balancing. Includes transmit and receive load balancing for **IPv4** traffic. Receive load balancing is achieved through **ARP** negotiation. This mode is only suitable for local addresses known to the kernel bonding module and therefore cannot be used behind a bridge with virtual machines.

The following types of link monitoring can be selected from the **Link Monitoring** drop-down list. It is a good idea to test which channel bonding module parameters work best for your bonded interfaces.

MII (Media Independent Interface)

The state of the carrier wave of the interface is monitored. This can be done by querying the driver, by querying MII registers directly, or by using **ethtool** to query the device. Three options are available:

Monitoring Frequency

The time interval, in milliseconds, between querying the driver or MII registers.

Link up delay

The time in milliseconds to wait before attempting to use a link that has been reported as up. This delay can be used if some gratuitous **ARP** requests are lost in the period immediately following the link being reported as "up". This can happen during switch initialization for example.

Link down delay

The time in milliseconds to wait before changing to another link when a previously active link has been reported as "down". This delay can be used if an attached switch takes a relatively long time to change to backup mode.

ARP

The address resolution protocol (**ARP**) is used to probe one or more peers to determine how well the link-layer connections are working. It is dependent on the device driver providing the transmit start time and the last receive time.

Two options are available:

Monitoring Frequency

The time interval, in milliseconds, between sending **ARP** requests.

ARP targets

A comma separated list of **IP** addresses to send **ARP** requests to.

10.3.7. Establishing a VLAN Connection

You can use **NetworkManager** to create a VLAN using an existing interface. Currently, at time of writing, you can only make VLANs on Ethernet devices.

Procedure 10.11. Adding a New VLAN Connection

You can configure a VLAN connection by opening the **Network Connections** window, clicking **Add**, and selecting **VLAN** from the list.

1. Right-click on the **NetworkManager** applet icon in the Notification Area and click **Edit Connections**. The **Network Connections** window appears.
2. Click the **Add** button to open the selection list. Select **VLAN** and then click **Create**. The **Editing VLAN Connection 1** window appears.
3. On the **VLAN** tab, select the parent interface from the drop-down list you want to use for the VLAN connection.
4. Enter the VLAN ID
5. Enter a VLAN interface name. This is the name of the VLAN interface that will be created. For example, "eth0.1" or "vlan2". (Normally this is either the parent interface name plus "." and the VLAN ID, or "vlan" plus the VLAN ID.)
6. Review and confirm the settings and then click the **Apply** button.

7. Edit the VLAN-specific settings by referring to the *Configuring the VLAN Tab* description below

Procedure 10.12. Editing an Existing VLAN Connection

Follow these steps to edit an existing VLAN connection.

1. Right-click on the **NetworkManager** applet icon in the Notification Area and click **Edit Connections**. The **Network Connections** window appears.
2. Select the connection you want to edit and click the **Edit** button.
3. Select the **VLAN** tab.
4. Configure the connection name, auto-connect behavior, and availability settings.

Three settings in the **Editing** dialog are common to all connection types:

- **Connection name** – Enter a descriptive name for your network connection. This name will be used to list this connection in the **VLAN** section of the **Network Connections** window.
- **Connect automatically** – Check this box if you want **NetworkManager** to auto-connect to this connection when it is available. See [Section 10.2.3, “Connecting to a Network Automatically”](#) for more information.
- **Available to all users** – Check this box to create a connection available to all users on the system. Changing this setting may require **root** privileges. See [Section 10.2.4, “User and System Connections”](#) for details.

5. Edit the VLAN-specific settings by referring to the *Configuring the VLAN Tab* description below

Saving Your New (or Modified) Connection and Making Further Configurations

Once you have finished editing your VLAN connection, click the **Apply** button and **NetworkManager** will immediately save your customized configuration. Given a correct configuration, you can connect to your new or customized connection by selecting it from the **NetworkManager** Notification Area applet. See [Section 10.2.1, “Connecting to a Network”](#) for information on using your new or altered connection.

You can further configure an existing connection by selecting it in the **Network Connections** window and clicking **Edit** to return to the **Editing** dialog.

Then, to configure:

- IPv4 settings for the connection, click the **IPv4 Settings** tab and proceed to [Section 10.3.9.4, “Configuring IPv4 Settings”](#).

Configuring the VLAN Tab

If you have already added a new VLAN connection (see [Procedure 10.11, “Adding a New VLAN Connection”](#) for instructions), you can edit the **VLAN** tab to set the parent interface and the VLAN ID.

Parent Interface

A previously configured interface can be selected in the drop-down list.

VLAN ID

The identification number to be used to tag the VLAN network traffic.

VLAN interface name

The name of the VLAN interface that will be created. For example, "eth0.1" or "vlan2".

Cloned MAC address

Optionally sets an alternate MAC address to use for identifying the VLAN interface. This can be used to change the source MAC address for packets sent on this VLAN.

MTU

Optionally sets a Maximum Transmission Unit (MTU) size to be used for packets to be sent over the VLAN connection.

10.3.8. Establishing an IP-over-InfiniBand (IPoIB) Connection

You can use **NetworkManager** to create an InfiniBand connection.

Procedure 10.13. Adding a New InfiniBand Connection

You can configure an InfiniBand connection by opening the **Network Connections** window, clicking **Add**, and selecting **InfiniBand** from the list.

1. Right-click on the **NetworkManager** applet icon in the Notification Area and click **Edit Connections**. The **Network Connections** window appears.
2. Click the **Add** button to open the selection list. Select **InfiniBand** and then click **Create**. The **Editing InfiniBand Connection 1** window appears.
3. On the **InfiniBand** tab, select the transport mode from the drop-down list you want to use for the InfiniBand connection.
4. Enter the InfiniBand MAC address.
5. Review and confirm the settings and then click the **Apply** button.
6. Edit the InfiniBand-specific settings by referring to the *Configuring the InfiniBand Tab* description below .

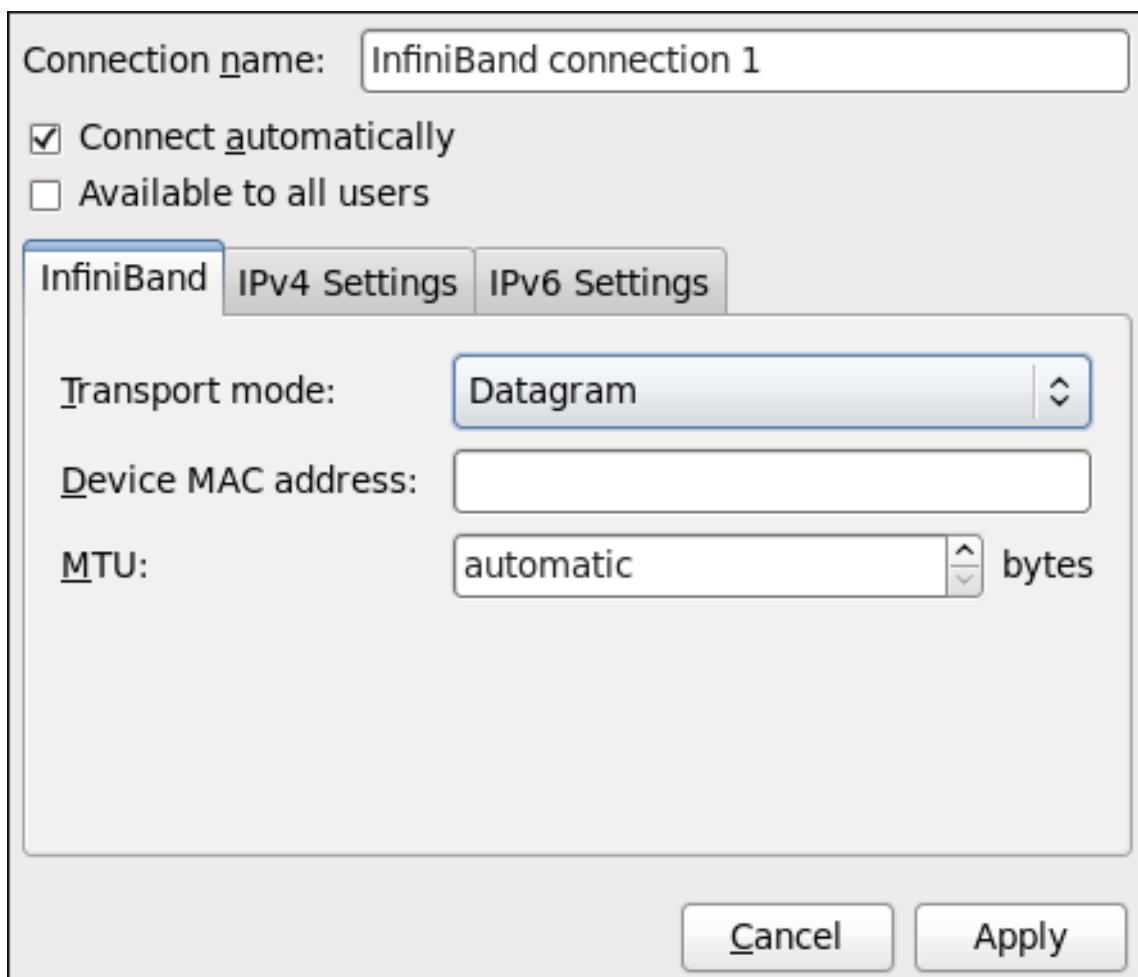


Figure 10.15. Editing the newly created InfiniBand connection 1

Procedure 10.14. Editing an Existing InfiniBand Connection

Follow these steps to edit an existing InfiniBand connection.

1. Right-click on the **NetworkManager** applet icon in the Notification Area and click **Edit Connections**. The **Network Connections** window appears.
2. Select the connection you want to edit and click the **Edit** button.
3. Select the **InfiniBand** tab.
4. Configure the connection name, auto-connect behavior, and availability settings.

Three settings in the **Editing** dialog are common to all connection types:

- **Connection name** – Enter a descriptive name for your network connection. This name will be used to list this connection in the **InfiniBand** section of the **Network Connections** window.
- **Connect automatically** – Check this box if you want **NetworkManager** to auto-connect to this connection when it is available. See [Section 10.2.3, “Connecting to a Network Automatically”](#) for more information.
- **Available to all users** – Check this box to create a connection available to all users on the system. Changing this setting may require **root** privileges. See [Section 10.2.4, “User and System Connections”](#) for details.

5. Edit the InfiniBand-specific settings by referring to the *Configuring the InfiniBand Tab* description below .

Saving Your New (or Modified) Connection and Making Further Configurations

Once you have finished editing your InfiniBand connection, click the **Apply** button and **NetworkManager** will immediately save your customized configuration. Given a correct configuration, you can connect to your new or customized connection by selecting it from the **NetworkManager** Notification Area applet. See [Section 10.2.1, “Connecting to a Network”](#) for information on using your new or altered connection.

You can further configure an existing connection by selecting it in the **Network Connections** window and clicking **Edit** to return to the **Editing** dialog.

Then, to configure:

- IPv4 settings for the connection, click the **IPv4 Settings** tab and proceed to [Section 10.3.9.4, “Configuring IPv4 Settings”](#); or,
- IPv6 settings for the connection, click the **IPv6 Settings** tab and proceed to [Section 10.3.9.5, “Configuring IPv6 Settings”](#).

Configuring the InfiniBand Tab

If you have already added a new InfiniBand connection (see [Procedure 10.13, “Adding a New InfiniBand Connection”](#) for instructions), you can edit the **InfiniBand** tab to set the parent interface and the InfiniBand ID.

Transport mode

Datagram or Connected mode can be selected from the drop-down list. Select the same mode the rest of your IPoIB network is using.

Device MAC address

The MAC address of the InfiniBand capable device to be used for the InfiniBand network traffic. This hardware address field will be pre-filled if you have InfiniBand hardware installed.

MTU

Optionally sets a Maximum Transmission Unit (MTU) size to be used for packets to be sent over the InfiniBand connection.

10.3.9. Configuring Connection Settings

10.3.9.1. Configuring 802.1X Security

802.1X security is the name of the IEEE standard for port-based Network Access Control (PNAC). Simply put, 802.1X security is a way of defining a *logical network* out of a physical one. All clients who want to join the logical network must authenticate with the server (a router, for example) using the correct 802.1X authentication method.

802.1X security is most often associated with securing wireless networks (WLANs), but can also be used to prevent intruders with physical access to the network (LAN) from gaining entry. In the past, DHCP servers were configured not to lease IP addresses to unauthorized users, but for various reasons this practice is both impractical and insecure, and thus is no longer recommended. Instead, 802.1X security is used to ensure a logically-secure network through port-based authentication.

802.1X provides a framework for WLAN and LAN access control and serves as an envelope for carrying one of the Extensible Authentication Protocol (EAP) types. An EAP type is a protocol that defines how WLAN security is achieved on the network.

You can configure 802.1X security for a wired or wireless connection type by opening the **Network Connections** window (see [Section 10.2.2, “Configuring New and Editing Existing Connections”](#)) and following the applicable procedure:

Procedure 10.15. For a wired connection...

1. Either click **Add**, select a new network connection for which you want to configure 802.1X security and then click **Create**, or select an existing connection and click **Edit**.
2. Then select the **802.1X Security** tab and check the **Use 802.1X security for this connection** check box to enable settings configuration.
3. Proceed to [Section 10.3.9.1.1, “Configuring TLS \(Transport Layer Security\) Settings”](#)

Procedure 10.16. For a wireless connection...

1. Either click on **Add**, select a new network connection for which you want to configure 802.1X security and then click **Create**, or select an existing connection and click **Edit**.
2. Select the **Wireless Security** tab.
3. Then click the **Security** dropdown and choose one of the following security methods: **LEAP**, **Dynamic WEP (802.1X)**, or **WPA & WPA2 Enterprise**
4. See [Section 10.3.9.1.1, “Configuring TLS \(Transport Layer Security\) Settings”](#) for descriptions of which EAP types correspond to your selection in the **Security** dropdown.

10.3.9.1.1. Configuring TLS (Transport Layer Security) Settings

With Transport Layer Security, the client and server mutually authenticate using the TLS protocol. The server demonstrates that it holds a digital certificate, the client proves its own identity using its client-side certificate, and key information is exchanged. Once authentication is complete, the TLS tunnel is no longer used. Instead, the client and server use the exchanged keys to encrypt data using AES, TKIP or WEP.

The fact that certificates must be distributed to all clients who want to authenticate means that the EAP-TLS authentication method is very strong, but also more complicated to set up. Using TLS security requires the overhead of a public key infrastructure (PKI) to manage certificates. The benefit of using TLS security is that a compromised password does not allow access to the (W)LAN: an intruder must also have access to the authenticating client's private key.

NetworkManager does not determine the version of TLS supported. **NetworkManager** gathers the parameters entered by the user and passes them to the daemon, **wpa_supplicant**, that handles the procedure. It in turn uses OpenSSL to establish the TLS tunnel. OpenSSL itself negotiates the SSL/TLS protocol version. It uses the highest version both ends support.

Identity

Identity string for EAP authentication methods, such as a user name or login name.

User certificate

Click to browse for, and select, a user's certificate.

CA certificate

Click to browse for, and select, a Certificate Authority's certificate.

Private key

Click to browse for, and select, a user's private key file. Note that the key must be password protected.

Private key password

Enter the user password corresponding to the user's private key.

10.3.9.1.2. Configuring Tunneled TLS Settings

Anonymous identity

This value is used as the unencrypted identity.

CA certificate

Click to browse for, and select, a Certificate Authority's certificate.

Inner authentication

PAP – Password Authentication Protocol.

MSCHAP – Challenge Handshake Authentication Protocol.

MSCHAPv2 – Microsoft Challenge Handshake Authentication Protocol version 2.

CHAP – Challenge Handshake Authentication Protocol.

Username

Enter the user name to be used in the authentication process.

Password

Enter the password to be used in the authentication process.

10.3.9.1.3. Configuring Protected EAP (PEAP) Settings

Anonymous Identity

This value is used as the unencrypted identity.

CA certificate

Click to browse for, and select, a Certificate Authority's certificate.

PEAP version

The version of Protected EAP to use. Automatic, 0 or 1.

Inner authentication

MSCHAPv2 – Microsoft Challenge Handshake Authentication Protocol version 2.

MD5 – Message Digest 5, a cryptographic hash function.

GTC – Generic Token Card.

Username

Enter the user name to be used in the authentication process.

Password

Enter the password to be used in the authentication process.

10.3.9.2. Configuring Wireless Security

Security

None – Do not encrypt the Wi-Fi connection.

WEP 40/128-bit Key – Wired Equivalent Privacy (WEP), from the IEEE 802.11 standard. Uses a single pre-shared key (PSK).

WEP 128-bit Passphrase – An MD5 hash of the passphrase will be used to derive a WEP key.

LEAP – Lightweight Extensible Authentication Protocol, from Cisco Systems.

Dynamic WEP (802.1X) – WEP keys are changed dynamically.

WPA & WPA2 Personal – Wi-Fi Protected Access (WPA), from the draft IEEE 802.11i standard. A replacement for WEP. Wi-Fi Protected Access II (WPA2), from the 802.11i-2004 standard. Personal mode uses a pre-shared key (WPA-PSK).

WPA & WPA2 Enterprise – WPA for use with a RADIUS authentication server to provide IEEE 802.1X network access control.

Password

Enter the password to be used in the authentication process.



NOTE

In the case of WPA and WPA2 (Personal and Enterprise), an option to select between Auto, WPA and WPA2 has been added. This option is intended for use with an access point that is offering both WPA and WPA2. Select one of the protocols if you would like to prevent roaming between the two protocols. Roaming between WPA and WPA2 on the same access point can cause loss of service.

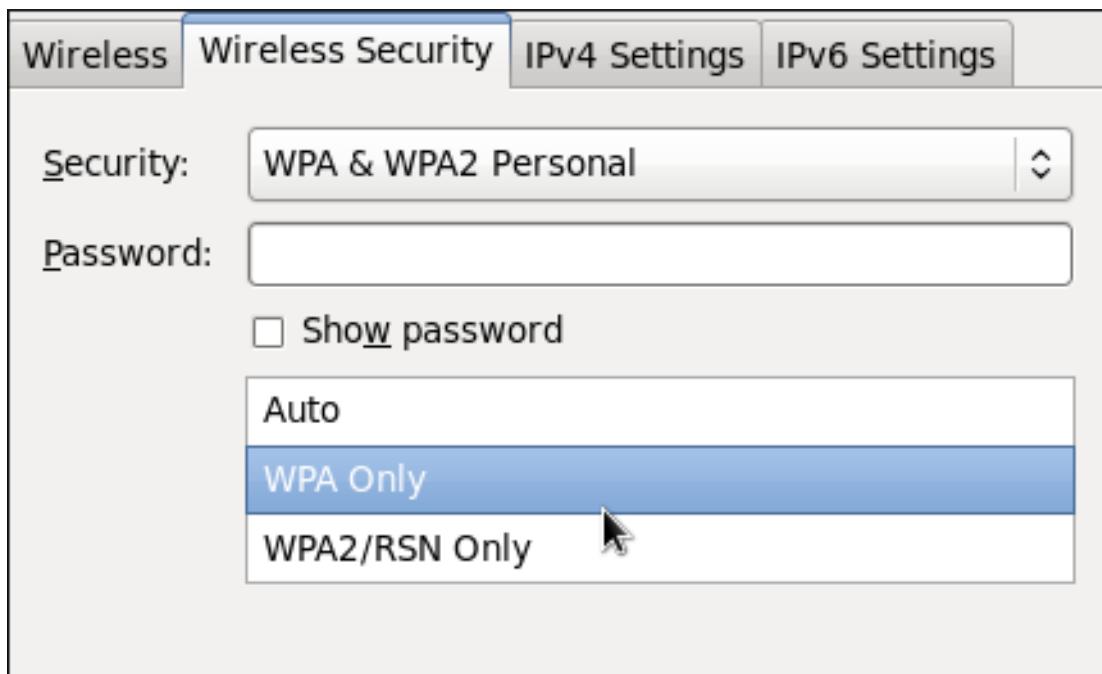


Figure 10.16. Editing the Wireless Security tab and selecting the WPA protocol

10.3.9.3. Configuring PPP (Point-to-Point) Settings

Configure Methods

Use point-to-point encryption (MPPE)

Microsoft Point-To-Point Encryption protocol (RFC 3078).

Allow BSD data compression

PPP BSD Compression Protocol (RFC 1977).

Allow Deflate data compression

PPP Deflate Protocol (RFC 1979).

Use TCP header compression

Compressing TCP/IP Headers for Low-Speed Serial Links (RFC 1144).

Send PPP echo packets

LCP Echo-Request and Echo-Reply Codes for loopback tests (RFC 1661).

10.3.9.4. Configuring IPv4 Settings

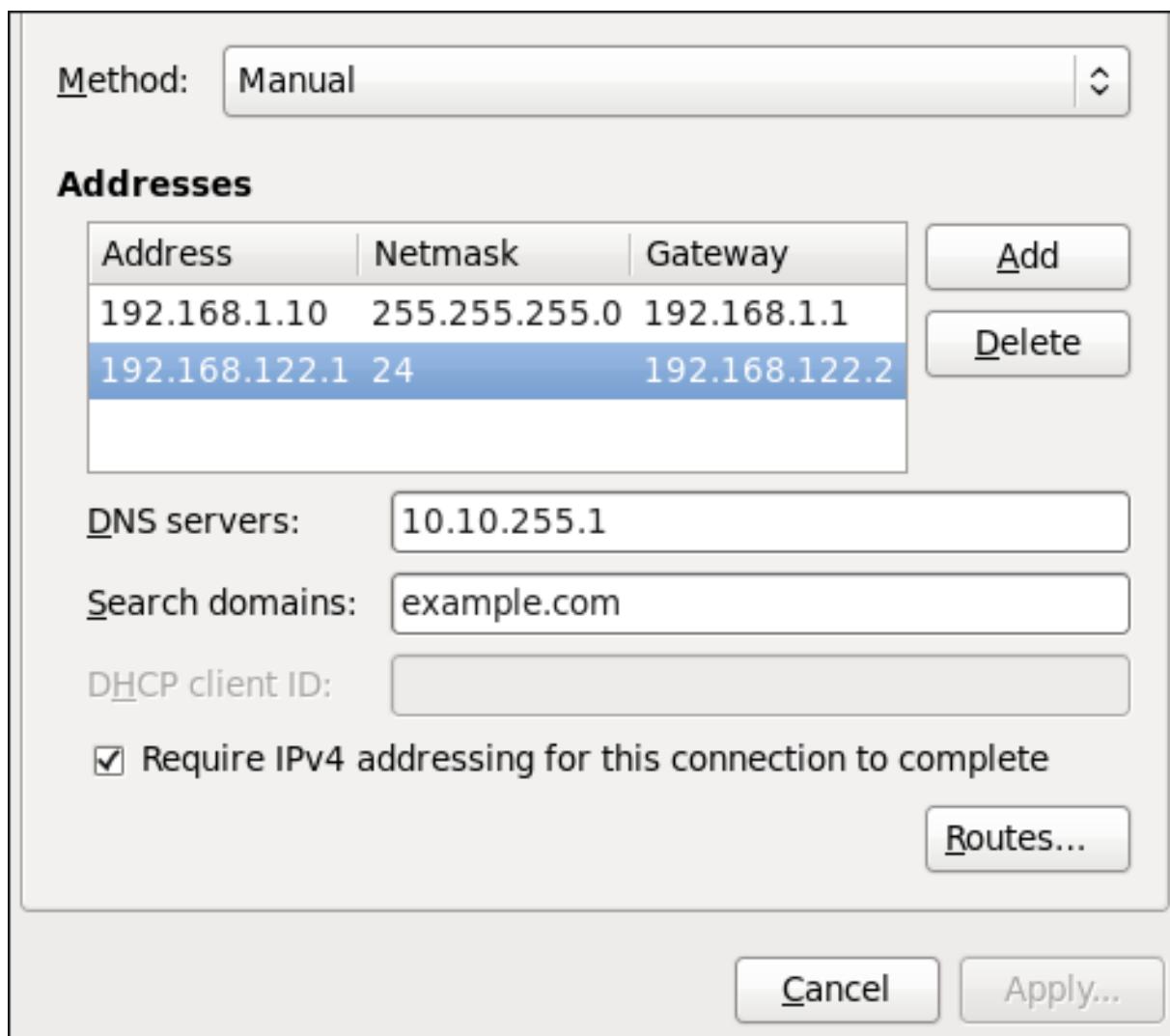


Figure 10.17. Editing the IPv4 Settings Tab

The **IPv4 Settings** tab allows you to configure the method by which you connect to the Internet and enter IP address, route, and DNS information as required. The **IPv4 Settings** tab is available when you create and modify one of the following connection types: wired, wireless, mobile broadband, VPN or DSL.

If you are using DHCP to obtain a dynamic IP address from a DHCP server, you can set **Method** to **Automatic (DHCP)**.

Setting the Method

Available IPv4 Methods by Connection Type

When you click the **Method** dropdown menu, depending on the type of connection you are configuring, you are able to select one of the following IPv4 connection methods. All of the methods are listed here according to which connection type or types they are associated with.

Method

Automatic (DHCP) – Choose this option if the network you are connecting to uses a DHCP server to assign IP addresses. You do not need to fill in the **DHCP client ID** field.

Automatic (DHCP) addresses only – Choose this option if the network you are connecting to uses a DHCP server to assign IP addresses but you want to assign DNS servers manually.

Link-Local Only – Choose this option if the network you are connecting to does not have a DHCP server and you do not want to assign IP addresses manually. Random addresses will be selected as per RFC 3927.

Shared to other computers – Choose this option if the interface you are configuring is for sharing an Internet or WAN connection.

Wired, Wireless and DSL Connection Methods

Manual – Choose this option if the network you are connecting to does not have a DHCP server and you want to assign IP addresses manually.

Mobile Broadband Connection Methods

Automatic (PPP) – Choose this option if the network you are connecting to uses a DHCP server to assign IP addresses.

Automatic (PPP) addresses only – Choose this option if the network you are connecting to uses a DHCP server to assign IP addresses but you want to assign DNS servers manually.

VPN Connection Methods

Automatic (VPN) – Choose this option if the network you are connecting to uses a DHCP server to assign IP addresses.

Automatic (VPN) addresses only – Choose this option if the network you are connecting to uses a DHCP server to assign IP addresses but you want to assign DNS servers manually.

DSL Connection Methods

Automatic (PPPoE) – Choose this option if the network you are connecting to uses a DHCP server to assign IP addresses.

Automatic (PPPoE) addresses only – Choose this option if the network you are connecting to uses a DHCP server to assign IP addresses but you want to assign DNS servers manually.

PPPoE Specific Configuration Steps

If more than one NIC is installed, and PPPoE will only be run over one NIC but not the other, then for correct PPPoE operation it is also necessary to lock the connection to the specific Ethernet device PPPoE is supposed to be run over. To lock the connection to one specific NIC, do **one** of the following:

- Enter the MAC address in **nm-connection-editor** for that connection. Optionally select **Connect automatically** and **Available to all users** to make the connection come up without requiring user login after system start.
- Set the hardware-address in the [802-3-ethernet] section in the appropriate file for that connection in **/etc/NetworkManager/system-connections** as follows:

```
[802-3-ethernet]
mac-address=00:11:22:33:44:55
```

Mere presence of the file in **/etc/NetworkManager/system-connections** means that it is “available to all users”. Ensure that **autoconnect=true** appears in the [connection] section for the connection to be brought up without requiring user login after system start.

For information on configuring static routes for the network connection, go to [Section 10.3.9.6, “Configuring Routes”](#).

10.3.9.5. Configuring IPv6 Settings

Method

Ignore – Choose this option if you want to disable IPv6 settings.

Automatic – Choose this option if the network you are connecting to uses a DHCP server to assign IP addresses.

Automatic, addresses only – Choose this option if the network you are connecting to uses a DHCP server to assign IP addresses but you want to assign DNS servers manually.

Manual – Choose this option if the network you are connecting to does not have a DHCP server and you want to assign IP addresses manually.

Link-Local Only – Choose this option if the network you are connecting to does not have a DHCP server and you do not want to assign IP addresses manually. Random addresses will be selected as per RFC 4862.

Shared to other computers – Choose this option if the interface you are configuring is for sharing an Internet or WAN connection.

Addresses

DNS servers – Enter a comma separated list of DNS servers.

Search domains – Enter a comma separated list of domain controllers.

For information on configuring static routes for the network connection, go to [Section 10.3.9.6, "Configuring Routes"](#).

10.3.9.6. Configuring Routes

A host's routing table will be automatically populated with routes to directly connected networks. The routes are learned by observing the network interfaces when they are "up". This section is for entering static routes to networks or hosts which can be reached by traversing an intermediate network or connection, such as a VPN or leased line.

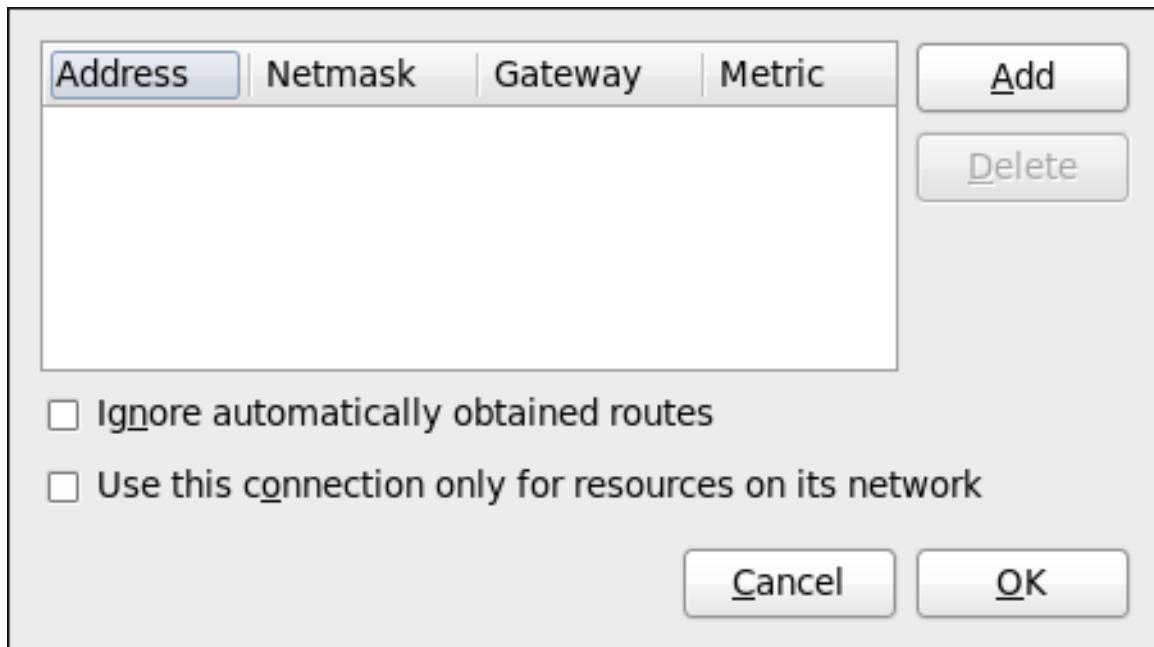


Figure 10.18. Configuring static network routes

Addresses

Address – The IP address of a network, sub-net or host.

Netmask – The netmask or prefix length of the IP address just entered.

Gateway – The IP address of the gateway leading to the network, sub-net or host.

Metric – A network cost, that is to say a preference value to give to this route. Lower values will be preferred over higher values.

Ignore automatically obtained routes

Select this check box to only use manually entered routes for this connection.

Use this connection only for resources on its network

Select this check box to prevent the connection from becoming the default route. Typical examples are where a connection is a VPN or a leased line to a head office and you do not want any Internet bound traffic to pass over the connection. Selecting this option means that only traffic specifically destined for routes learned automatically over the connection or entered here manually will be routed over the connection.

CHAPTER 11. NETWORK INTERFACES

Under Red Hat Enterprise Linux, all network communications occur between configured software *interfaces* and *physical networking devices* connected to the system.

The configuration files for network interfaces are located in the **/etc/sysconfig/network-scripts/** directory. The scripts used to activate and deactivate these network interfaces are also located here. Although the number and type of interface files can differ from system to system, there are three categories of files that exist in this directory:

1. *Interface configuration files*
2. *Interface control scripts*
3. *Network function files*

The files in each of these categories work together to enable various network devices.

This chapter explores the relationship between these files and how they are used.

11.1. NETWORK CONFIGURATION FILES

Before delving into the interface configuration files, let us first itemize the primary configuration files used in network configuration. Understanding the role these files play in setting up the network stack can be helpful when customizing a Red Hat Enterprise Linux system.

The primary network configuration files are as follows:

/etc/hosts

The main purpose of this file is to resolve host names that cannot be resolved any other way. It can also be used to resolve host names on small networks with no **DNS** server. Regardless of the type of network the computer is on, this file should contain a line specifying the **IP** address of the loopback device (**127.0.0.1**) as **localhost.localdomain**. For more information, see the **hosts(5)** manual page.

/etc/resolv.conf

This file specifies the **IP** addresses of **DNS** servers and the search domain. Unless configured to do otherwise, the network initialization scripts populate this file. For more information about this file, see the **resolv.conf(5)** manual page.

/etc/sysconfig/network

This file specifies routing and host information for all network interfaces. It is used to contain directives which are to have global effect and not to be interface specific. For more information about this file and the directives it accepts, see [Section D.1.14, “/etc/sysconfig/network”](#).

/etc/sysconfig/network-scripts/ifcfg-*interface-name*

For each network interface, there is a corresponding interface configuration script. Each of these files provide information specific to a particular network interface. See [Section 11.2, “Interface Configuration Files”](#) for more information on this type of file and the directives it accepts.



IMPORTANT

Network interface names may be different on different hardware types. See [Appendix A, Consistent Network Device Naming](#) for more information.



WARNING

The `/etc/sysconfig/networking/` directory is used by the now deprecated **Network Administration Tool (system-config-network)**. Its contents should **not** be edited manually. Using only one method for network configuration is strongly encouraged, due to the risk of configuration deletion. For more information about configuring network interfaces using graphical configuration tools, see [Chapter 10, NetworkManager](#).

11.1.1. Setting the Host Name

To permanently change the static host name, change the `HOSTNAME` directive in the `/etc/sysconfig/network` file. For example:

```
HOSTNAME=penguin.example.com
```

Red Hat recommends the static host name matches the *fully qualified domain name* (FQDN) used for the machine in DNS, such as `host.example.com`. It is also recommended that the static host name consists only of 7 bit ASCII lower-case characters, no spaces or dots, and limits itself to the format allowed for DNS domain name labels, even though this is not a strict requirement. Older specifications do not permit the underscore, and so their use is not recommended. Changes will only take effect when the networking service, or the system, is restarted.

Note that the FQDN of the host can be supplied by a DNS resolver, by settings in `/etc/sysconfig/network`, or by the `/etc/hosts` file. The default setting of **hosts: files dns** in `/etc/nsswitch.conf` causes the configuration files to be checked before a resolver. The default setting of **multi on** in the `/etc/host.conf` file means that all valid values in the `/etc/hosts` file are returned, not just the first.

Sometimes you may need to use the host table in the `/etc/hosts` file instead of the `HOSTNAME` directive in `/etc/sysconfig/network`, for example, when DNS is not running during system bootup.

To change the host name using the `/etc/hosts` file, add lines to it in the following format:

```
192.168.1.2 penguin.example.com penguin
```

11.2. INTERFACE CONFIGURATION FILES

Interface configuration files control the software interfaces for individual network devices. As the system boots, it uses these files to determine what interfaces to bring up and how to configure them. These files are usually named **`ifcfg-name`**, where *name* refers to the name of the device that the configuration file controls.

11.2.1. Ethernet Interfaces

One of the most common interface files is **/etc/sysconfig/network-scripts/ifcfg-eth0**, which controls the first Ethernet *network interface card* or NIC in the system. In a system with multiple NICs, there are multiple **ifcfg-ethX** files (where *X* is a unique number corresponding to a specific interface). Because each device has its own configuration file, an administrator can control how each interface functions individually.

The following is a sample **ifcfg-eth0** file for a system using a fixed **IP** address:

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
NETMASK=255.255.255.0
IPADDR=10.0.1.27
USERCTL=no
```

The values required in an interface configuration file can change based on other values. For example, the **ifcfg-eth0** file for an interface using **DHCP** looks different because **IP** information is provided by the **DHCP** server:

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
```

NetworkManager is graphical configuration tool which provides an easy way to make changes to the various network interface configuration files (see [Chapter 10, NetworkManager](#) for detailed instructions on using this tool).

However, it is also possible to manually edit the configuration files for a given network interface.

Below is a listing of the configurable parameters in an Ethernet interface configuration file:

BONDING_OPTS=parameters

sets the configuration parameters for the bonding device, and is used in **/etc/sysconfig/network-scripts/ifcfg-bond*N*** (see [Section 11.2.4, “Channel Bonding Interfaces”](#)). These parameters are identical to those used for bonding devices in **/sys/class/net/bonding_device/bonding**, and the module parameters for the bonding driver as described in **bonding** *Module Directives*.

This configuration method is used so that multiple bonding devices can have different configurations. In Red Hat Enterprise Linux 6, place all interface-specific bonding options after the **BONDING_OPTS** directive in **ifcfg-name** files. See [Where to specify bonding module parameters](#) for more information.

BOOTPROTO=protocol

where *protocol* is one of the following:

- **none** – No boot-time protocol should be used.
- **bootp** – The **BOOTP** protocol should be used.
- **dhcp** – The **DHCP** protocol should be used.

BROADCAST=address

where *address* is the broadcast address. This directive is deprecated, as the value is calculated automatically with **ipcalc**.

DEVICE=name

where *name* is the name of the physical device (except for dynamically-allocated **PPP** devices where it is the *logical name*).

DHCP_HOSTNAME=name

where *name* is a short host name to be sent to the **DHCP** server. Use this option only if the **DHCP** server requires the client to specify a host name before receiving an **IP** address.

DHCPV6C=answer

where *answer* is one of the following:

- **yes** – Use **DHCP** to obtain an **IPv6** address for this interface.
- **no** – Do not use **DHCP** to obtain an **IPv6** address for this interface. This is the default value.

An **IPv6** link-local address will still be assigned by default. The link-local address is based on the MAC address of the interface as per *RFC 4862*.

DHCPV6C_OPTIONS=answer

where *answer* is one of the following:

- **-P** – Enable **IPv6** prefix delegation.
- **-S** – Use **DHCP** to obtain stateless configuration only, not addresses, for this interface.
- **-N** – Restore normal operation after using the **-T** or **-P** options.
- **-T** – Use **DHCP** to obtain a temporary **IPv6** address for this interface.
- **-D** – Override the default when selecting the type of *DHCP Unique Identifier* (DUID) to use.

By default, the **DHCP** client (`dhclient`) creates a *DHCP Unique Identifier* (DUID) based on the link-layer address (DUID-LL) if it is running in stateless mode (with the **-S** option, to not request an address), or it creates an identifier based on the link-layer address plus a timestamp (DUID-LLT) if it is running in stateful mode (without **-S**, requesting an address). The **-D** option overrides this default, with a value of either **LL** or **LLT**.

DNS{1,2}=address

where *address* is a name server address to be placed in **/etc/resolv.conf** provided that the **PEERDNS** directive is not set to **no**.

ETHTOOL_OPTS=options

where *options* are any device-specific options supported by **ethtool**. For example, if you wanted to force 100Mb, full duplex:

```
ETHTOOL_OPTS="autoneg off speed 100 duplex full"
```

Instead of a custom initscript, use **ETHTOOL_OPTS** to set the interface speed and duplex settings. Custom initscripts run outside of the network init script lead to unpredictable results during a post-boot network service restart.



IMPORTANT

Changing speed or duplex settings almost always requires disabling auto-negotiation with the **autoneg off** option. This option needs to be stated first, as the option entries are order-dependent.

See [Section 11.8, “Ehtool”](#) for more **ethtool** options.

HOTPLUG=answer

where *answer* is one of the following:

- **yes** – This device should be activated when it is hot-plugged (this is the default option).
- **no** – This device should *not* be activated when it is hot-plugged.

The **HOTPLUG=no** option can be used to prevent a channel bonding interface from being activated when a bonding kernel module is loaded.

See [Section 11.2.4, “Channel Bonding Interfaces”](#) for more information about channel bonding interfaces.

HWADDR=MAC-address

where *MAC-address* is the hardware address of the Ethernet device in the form *AA:BB:CC:DD:EE:FF*. This directive must be used in machines containing more than one NIC to ensure that the interfaces are assigned the correct device names regardless of the configured load order for each NIC's module. This directive should **not** be used in conjunction with **MACADDR**.



NOTE

- Persistent device names are now handled by [**/etc/udev/rules.d/70-persistent-net.rules**](#).
- **HWADDR** must not be used with System z network devices.
- See Section 25.3.3, “Mapping subchannels and network device names”, in the [*Red Hat Enterprise Linux 6 Installation Guide*](#).

IPADDRn=address

where *address* is the **IPv4** address and the *n* is expected to be consecutive positive integers starting from 0 (for example, IPADDR0). It is used for configurations with multiple IP addresses on an interface. It can be omitted if there is only one address being configured.

IPV6ADDR=address

where *address* is the first static, or primary, **IPv6** address on an interface.

The format is Address/Prefix-length. If no prefix length is specified, /64 is assumed. Note that this setting depends on **IPV6INIT** being enabled.

IPV6ADDR_SECONDARIES=address

where *address* is one or more, space separated, additional **IPv6** addresses.

The format is Address/Prefix-length. If no prefix length is specified, /64 is assumed. Note that this setting depends on **IPV6INIT** being enabled.

IPV6INIT=answer

where *answer* is one of the following:

- **yes** – Initialize this interface for **IPv6** addressing.
- **no** – Do not initialize this interface for **IPv6** addressing. This is the default value.

This setting is required for **IPv6** static and **DHCP** assignment of **IPv6** addresses. It does not affect *IPv6 Stateless Address Autoconfiguration* (SLAAC) as per [RFC 4862](#).

See [Section D.1.14, "/etc/sysconfig/network"](#) for information on disabling **IPv6**.

IPV6_AUTOCONF=answer

where *answer* is one of the following:

- **yes** – Enable **IPv6** autoconf configuration for this interface.
- **no** – Disable **IPv6** autoconf configuration for this interface.

If enabled, an **IPv6** address will be requested using *Neighbor Discovery* (ND) from a router running the **radvd** daemon.

Note that the default value of **IPV6_AUTOCONF** depends on **IPV6FORWARDING** as follows:

- If **IPV6FORWARDING=yes**, then **IPV6_AUTOCONF** will default to **no**.
- If **IPV6FORWARDING=no**, then **IPV6_AUTOCONF** will default to **yes** and **IPV6_ROUTER** has no effect.

IPV6_MTU=value

where *value* is an optional dedicated MTU for this interface.

IPV6_PRIVACY=rfc3041

where *rfc3041* optionally sets this interface to support [RFC 3041 Privacy Extensions for Stateless Address Autoconfiguration in IPv6](#). Note that this setting depends on **IPV6INIT** option being enabled.

The default is for *RFC 3041* support to be disabled. Stateless Autoconfiguration will derive addresses based on the MAC address, when available, using the modified **EUI-64** method. The address is appended to a prefix but as the address is normally derived from the MAC address it is globally unique even when the prefix changes. In the case of a link-local address the prefix is **fe80::/64** as per [RFC 2462 IPv6 Stateless Address Autoconfiguration](#).

LINKDELAY=time

where *time* is the number of seconds to wait for link negotiation before configuring the device. The default is 5 secs. Delays in link negotiation, caused by **STP** for example, can be overcome by increasing this value.

MACADDR=MAC-address

where *MAC-address* is the hardware address of the Ethernet device in the form *AA:BB:CC:DD:EE:FF*.

This directive is used to assign a MAC address to an interface, overriding the one assigned to the physical NIC. This directive should **not** be used in conjunction with the **HWADDR** directive.

MASTER=bond-interface

where *bond-interface* is the channel bonding interface to which the Ethernet interface is linked.

This directive is used in conjunction with the **SLAVE** directive.

See [Section 11.2.4, “Channel Bonding Interfaces”](#) for more information about channel bonding interfaces.

NETMASKn=mask

where *mask* is the netmask value and the *n* is expected to be consecutive positive integers starting from 0 (for example, NETMASK0). It is used for configurations with multiple IP addresses on an interface. It can be omitted if there is only one address being configured.

NETWORK=address

where *address* is the network address. This directive is deprecated, as the value is calculated automatically with **ipcalc**.

NM_CONTROLLED=answer

where *answer* is one of the following:

- **yes** – **NetworkManager** is permitted to configure this device. This is the default behavior and can be omitted.
- **no** – **NetworkManager** is not permitted to configure this device.



NOTE

The **NM_CONTROLLED** directive is now, as of Red Hat Enterprise Linux 6.3, dependent on the **NM_BOND_VLAN_ENABLED** directive in **/etc/sysconfig/network**. If and only if that directive is present and is one of **yes**, **y**, or **true**, will **NetworkManager** detect and manage bonding and VLAN interfaces.

ONBOOT=answer

where *answer* is one of the following:

- **yes** – This device should be activated at boot-time.
- **no** – This device should not be activated at boot-time.

PEERDNS=answer

where *answer* is one of the following:

- **yes** – Modify **/etc/resolv.conf** if the **DNS** directive is set, if using **DHCP**, or if using Microsoft's [RFC 1877](#) **IPCP** extensions with **PPP**. In all cases **yes** is the default.
- **no** – Do not modify **/etc/resolv.conf**.

SLAVE=answer

where *answer* is one of the following:

- **yes** – This device is controlled by the channel bonding interface specified in the **MASTER** directive.
- **no** – This device is *not* controlled by the channel bonding interface specified in the **MASTER** directive.

This directive is used in conjunction with the **MASTER** directive.

See [Section 11.2.4, “Channel Bonding Interfaces”](#) for more about channel bonding interfaces.

SRCADDR=address

where *address* is the specified source **IP** address for outgoing packets.

USERCTL=answer

where *answer* is one of the following:

- **yes** – Non-**root** users are allowed to control this device.
- **no** – Non-**root** users are not allowed to control this device.

11.2.2. Specific ifcfg Options for Linux on System z

SUBCHANNELS=<read_device_bus_id>, <write_device_bus_id>, <data_device_bus_id>

where <*read_device_bus_id*>, <*write_device_bus_id*>, and <*data_device_bus_id*> are the three device bus IDs representing a network device.

PORNAME=myname;

where *myname* is the Open Systems Adapter (OSA) portname or LAN Channel Station (LCS) portnumber.

CTCPROT=answer

where *answer* is one of the following:

- **0** – Compatibility mode, TCP/IP for Virtual Machines (used with non-Linux peers other than IBM S/390 and IBM System z operating systems). This is the default mode.
- **1** – Extended mode, used for Linux-to-Linux Peers.
- **3** – Compatibility mode for S/390 and IBM System z operating systems.

This directive is used in conjunction with the NETTYPE directive. It specifies the CTC protocol for NETTYPE='ctc'. The default is 0.

OPTION='answer'

where '*answer*' is a quoted string of any valid sysfs attributes and their value. The Red Hat Enterprise Linux installer currently uses this to configure the layer mode, (layer2), and the relative port number, (portno), of QETH devices. For example:

OPTIONS='layer2=1 portno=0'

11.2.3. Required ifcfg Options for Linux on System z

NETTYPE=answer

where *answer* is one of the following:

- **ctc** – Channel-to-Channel communication. For point-to-point TCP/IP or TTY.
- **lcs** – LAN Channel Station (LCS).
- **qeth** – QETH (QDIO Ethernet). This is the default network interface. It is the preferred installation method for supporting real or virtual OSA cards and HiperSockets devices.

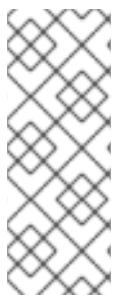
11.2.4. Channel Bonding Interfaces

Red Hat Enterprise Linux allows administrators to bind multiple network interfaces together into a single channel using the **bonding** kernel module and a special network interface called a *channel bonding interface*. Channel bonding enables two or more network interfaces to act as one, simultaneously increasing the bandwidth and providing redundancy.



WARNING

The use of direct cable connections without network switches is not supported for bonding. The failover mechanisms described here will not work as expected without the presence of network switches. See the Red Hat Knowledgebase article [Why is bonding in not supported with direct connection using crossover cables?](#) for more information.



NOTE

The active-backup, balance-tlb and balance-alb modes do not require any specific configuration of the switch. Other bonding modes require configuring the switch to aggregate the links. For example, a Cisco switch requires EtherChannel for Modes 0, 2, and 3, but for Mode 4 LACP and EtherChannel are required. See the documentation supplied with your switch and the **bonding.txt** file in the `kernel-doc` package (see [Section 31.9, “Additional Resources”](#)).

11.2.4.1. Check if Bonding Kernel Module is Installed

In Red Hat Enterprise Linux 6, the bonding module is not loaded by default. You can load the module by issuing the following command as **root**:

```
~]# modprobe --first-time bonding
```

No visual output indicates the module was not running and has now been loaded. This activation will not persist across system restarts. See [Section 31.7, “Persistent Module Loading”](#) for an explanation of persistent module loading. Note that given a correct configuration file using the **BONDING_OPTS** directive, the bonding module will be loaded as required and therefore does not need to be loaded separately.

To display information about the module, issue the following command:

```
~]$ modinfo bonding
```

See the **modprobe(8)** man page for more command options and see [Chapter 31, Working with Kernel Modules](#) for information on loading and unloading modules.

11.2.4.2. Create a Channel Bonding Interface

To create a channel bonding interface, create a file in the **/etc/sysconfig/network-scripts/** directory called **ifcfg-bondN**, replacing *N* with the number for the interface, such as **0**.

The contents of the file can be identical to whatever type of interface is getting bonded, such as an Ethernet interface. The only difference is that the **DEVICE** directive is **bondN**, replacing *N* with the number for the interface. The **NM_CONTROLLED** directive can be added to prevent **NetworkManager** from configuring this device.

Example 11.1. Example ifcfg-bond0 interface configuration file

The following is an example of a channel bonding interface configuration file:

```
DEVICE=bond0
IPADDR=192.168.1.1
NETMASK=255.255.255.0
ONBOOT=yes
BOOTPROTO=none
USERCTL=no
NM_CONTROLLED=no
BONDING_OPTS="bonding parameters separated by spaces"
```

The MAC address of the bond will be taken from the first interface to be enslaved. It can also be specified using the **HWADDR** directive if required. If you want **NetworkManager** to control this interface, remove the **NM_CONTROLLED=no** directive, or set it to **yes**, and add **TYPE=Bond** and **BONDING_MASTER=yes**.

After the channel bonding interface is created, the network interfaces to be bound together must be configured by adding the **MASTER** and **SLAVE** directives to their configuration files. The configuration files for each of the channel-bonded interfaces can be nearly identical.

Example 11.2. Example ifcfg-ethX bonded interface configuration file

If two Ethernet interfaces are being channel bonded, both **eth0** and **eth1** can be as follows:

```
DEVICE=ethX
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes
USERCTL=no
NM_CONTROLLED=no
```

In this example, replace *X* with the numerical value for the interface.

Once the interfaces have been configured, restart the network service to bring the bond up. As **root**, issue the following command:

```
~]# service network restart
```

To view the status of a bond, view the **/proc** file by issuing a command in the following format:

```
cat /proc/net/bonding/bondN
```

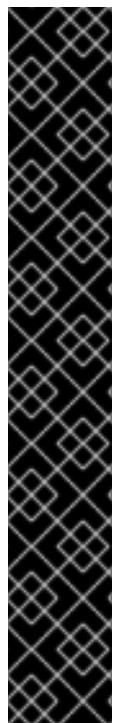
For example:

```
~]$ cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.6.0 (September 26, 2009)

Bonding Mode: load balancing (round-robin)
MII Status: down
MII Polling Interval (ms): 0
Up Delay (ms): 0
Down Delay (ms): 0
```

For further instructions and advice on configuring the bonding module and to view the list of bonding parameters, see [Section 31.8.1, “Using Channel Bonding”](#).

Support for bonding was added to **NetworkManager** in Red Hat Enterprise Linux 6.3. See [Section 11.2.1, “Ethernet Interfaces”](#) for an explanation of **NM_CONTROLLED** and the **NM_BOND_VLAN_ENABLED** directive.



IMPORTANT

In Red Hat Enterprise Linux 6, interface-specific parameters for the bonding kernel module must be specified as a space-separated list in the **BONDING_OPTS="bonding parameters"** directive in the **ifcfg-bondN** interface file. Do **not** specify options specific to a bond in **/etc/modprobe.d/bonding.conf**, or in the deprecated **/etc/modprobe.conf** file.

The **max_bonds** parameter is not interface specific and therefore, if required, should be specified in **/etc/modprobe.d/bonding.conf** as follows:

```
options bonding max_bonds=1
```

However, the **max_bonds** parameter should **not** be set when using **ifcfg-bondN** files with the **BONDING_OPTS** directive as this directive will cause the network scripts to create the bond interfaces as required.

Note that any changes to **/etc/modprobe.d/bonding.conf** will not take effect until the module is next loaded. A running module must first be unloaded. See [Chapter 31, Working with Kernel Modules](#) for more information on loading and unloading modules.

11.2.4.2.1. Creating Multiple Bonds

In Red Hat Enterprise Linux 6, for each bond a channel bonding interface is created including the **BONDING_OPTS** directive. This configuration method is used so that multiple bonding devices can have different configurations. To create multiple channel bonding interfaces, proceed as follows:

- Create multiple **ifcfg-bondN** files with the **BONDING_OPTS** directive; this directive will cause the network scripts to create the bond interfaces as required.
- Create, or edit existing, interface configuration files to be bonded and include the **SLAVE** directive.
- Assign the interfaces to be bonded, the slave interfaces, to the channel bonding interfaces by means of the **MASTER** directive.

Example 11.3. Example multiple ifcfg-bondN interface configuration files

The following is an example of a channel bonding interface configuration file:

```
DEVICE=bondN
IPADDR=192.168.1.1
NETMASK=255.255.255.0
ONBOOT=yes
BOOTPROTO=none
USERCTL=no
NM_CONTROLLED=no
BONDING_OPTS="bonding parameters separated by spaces"
```

In this example, replace *N* with the number for the bond interface. For example, to create two bonds create two configuration files, **ifcfg-bond0** and **ifcfg-bond1**.

Create the interfaces to be bonded as per [Example 11.2, “Example ifcfg-ethX bonded interface configuration file”](#) and assign them to the bond interfaces as required using the **MASTER=bond*N*** directive. For example, continuing on from the example above, if two interfaces per bond are required, then for two bonds create four interface configuration files and assign the first two using **MASTER=bond0** and the next two using **MASTER=bond1**.

11.2.5. Configuring a VLAN over a Bond

This section will show configuring a VLAN over a bond consisting of two Ethernet links between a server and an Ethernet switch. The switch has a second bond to another server. Only the configuration for the first server will be shown as the other is essentially the same apart from the IP addresses.



WARNING

The use of direct cable connections without network switches is not supported for bonding. The failover mechanisms described here will not work as expected without the presence of network switches. See the Red Hat Knowledgebase article [Why is bonding in not supported with direct connection using crossover cables?](#) for more information.



NOTE

The active-backup, balance-tlb and balance-alb modes do not require any specific configuration of the switch. Other bonding modes require configuring the switch to aggregate the links. For example, a Cisco switch requires EtherChannel for Modes 0, 2, and 3, but for Mode 4 LACP and EtherChannel are required. See the documentation supplied with your switch and the **bonding.txt** file in the kernel-doc package (see [Section 31.9, "Additional Resources"](#)).

Check the available interfaces on the server:

```
~]$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast state DOWN qlen 1000
    link/ether 52:54:00:19:28:fe brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast state DOWN qlen 1000
    link/ether 52:54:00:f6:63:9a brd ff:ff:ff:ff:ff:ff
```

Procedure 11.1. Configuring the Interfaces on the Server

- Configure a slave interface using **eth0**:

```
~]# vi /etc/sysconfig/network-scripts/ifcfg-eth0
NAME=bond0-slave0
DEVICE=eth0
TYPE=Ethernet
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes
NM_CONTROLLED=no
```

The use of the NAME directive is optional. It is for display by a GUI interface, such as **nm-connection-editor** and **nm-applet**.

- Configure a slave interface using **eth1**:

```
~]# vi /etc/sysconfig/network-scripts/ifcfg-eth1
NAME=bond0-slave1
DEVICE=eth1
TYPE=Ethernet
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes
NM_CONTROLLED=no
```

The use of the NAME directive is optional. It is for display by a GUI interface, such as **nm-connection-editor** and **nm-applet**.

3. Configure a channel bonding interface **ifcfg-bond0**:

```
~]# vi /etc/sysconfig/network-scripts/ifcfg-bond0
NAME=bond0
DEVICE=bond0
BONDING_MASTER=yes
TYPE=Bond
IPADDR=192.168.100.100
NETMASK=255.255.255.0
ONBOOT=yes
BOOTPROTO=none
BONDING_OPTS="mode=active-backup miimon=100"
NM_CONTROLLED=no
```

The use of the NAME directive is optional. It is for display by a GUI interface, such as **nm-connection-editor** and **nm-applet**. In this example MII is used for link monitoring, see the [Section 31.8.1.1, “Bonding Module Directives”](#) section for more information on link monitoring.

4. Check the status of the interfaces on the server:

```
~]$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            inet6 ::1/128 scope host
                valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
    qlen 1000
    link/ether 52:54:00:19:28:fe brd ff:ff:ff:ff:ff:ff
        inet6 fe80::5054:ff:fe19:28fe/64 scope link
            valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
    qlen 1000
    link/ether 52:54:00:f6:63:9a brd ff:ff:ff:ff:ff:ff
        inet6 fe80::5054:ff:fef6:639a/64 scope link
            valid_lft forever preferred_lft forever
```

Procedure 11.2. Resolving Conflicts with Interfaces

The interfaces configured as slaves should not have **IP** addresses assigned to them apart from the **IPv6** link-local addresses (starting **fe80**). If you have an unexpected **IP** address, then there may be another configuration file with ONBOOT set to **yes**.

1. If this occurs, issue the following command to list all **ifcfg** files that may be causing a conflict:

```
~]$ grep -r "ONBOOT=yes" /etc/sysconfig/network-scripts/ | cut -f1 -d":" | xargs grep -E
"IPADDR|SLAVE"
/etc/sysconfig/network-scripts/ifcfg-lo:IPADDR=127.0.0.1
```

The above shows the expected result on a new installation. Any file having both the ONBOOT directive as well as the IPADDR or SLAVE directive will be displayed. For example, if the **ifcfg-eth1** file was incorrectly configured, the display might look similar to the following:

```
~]$ grep -r "ONBOOT=yes" /etc/sysconfig/network-scripts/ | cut -f1 -d":" | xargs grep -E
"IPADDR|SLAVE"
```

```
/etc/sysconfig/network-scripts/ifcfg-lo:IPADDR=127.0.0.1
/etc/sysconfig/network-scripts/ifcfg-eth1:SLAVE=yes
/etc/sysconfig/network-scripts/ifcfg-eth1:IPADDR=192.168.55.55
```

- Any other configuration files found should be moved to a different directory for backup, or assigned to a different interface by means of the HWADDR directive. After resolving any conflict set the interfaces "down" and "up" again or restart the network service as **root**:

```
~]# service network restart
Shutting down interface bond0: [ OK ]
Shutting down loopback interface: [ OK ]
Bringing up loopback interface: [ OK ]
Bringing up interface bond0: Determining if ip address 192.168.100.100 is already in use for device bond0...
[ OK ]
```

If you are using **NetworkManager**, you might need to restart it at this point to make it forget the unwanted **IP** address. As **root**:

```
~]# service NetworkManager restart
```

Procedure 11.3. Checking the bond on the Server

- Bring up the bond on the server as **root**:

```
~]# ifup /etc/sysconfig/network-scripts/ifcfg-bond0
Determining if ip address 192.168.100.100 is already in use for device bond0...
```

- Check the status of the interfaces on the server:

```
~]$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            inet6 ::1/128 scope host
                valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    master bond0 state UP qlen 1000
    link/ether 52:54:00:19:28:fe brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    master bond0 state UP qlen 1000
    link/ether 52:54:00:f6:63:9a brd ff:ff:ff:ff:ff:ff
4: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue
    state UP
    link/ether 52:54:00:19:28:fe brd ff:ff:ff:ff:ff:ff
        inet 192.168.100.100/24 brd 192.168.100.255 scope global bond0
            inet6 fe80::5054:f6:63:9a/64 scope link
                valid_lft forever preferred_lft forever
```

Notice that **eth0** and **eth1** have **master bond0 state UP** and **bond0** has status of **MASTER,UP**.

- View the bond configuration details:

```
~]$ cat /proc/net/bonding/bond0
```

```
Ethernet Channel Bonding Driver: v3.6.0 (September 26, 2009)
```

```
Bonding Mode: transmit load balancing
Primary Slave: None
Currently Active Slave: eth0
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0
```

```
Slave Interface: eth0
MII Status: up
Speed: 100 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 52:54:00:19:28:fe
Slave queue ID: 0
```

```
Slave Interface: eth1
MII Status: up
Speed: 100 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 52:54:00:f6:63:9a
Slave queue ID: 0
```

4. Check the routes on the server:

```
~]$ ip route
192.168.100.0/24 dev bond0 proto kernel scope link src 192.168.100.100
169.254.0.0/16 dev bond0 scope link metric 1004
```

Procedure 11.4. Configuring the VLAN on the Server



IMPORTANT

At the time of writing, it is important that the bond has slaves and that they are "up" before bringing up the VLAN interface. At the time of writing, adding a VLAN interface to a bond without slaves does not work. In Red Hat Enterprise Linux 6, setting the `ONPARENT` directive to `yes` is important to ensure that the VLAN interface does not attempt to come up before the bond is up. This is because a VLAN virtual device takes the MAC address of its parent, and when a NIC is enslaved, the bond changes its MAC address to that NIC's MAC address.



NOTE

A VLAN slave cannot be configured on a bond with the **fail_over_mac=follow** option, because the VLAN virtual device cannot change its MAC address to match the parent's new MAC address. In such a case, traffic would still be sent with the now incorrect source MAC address.

Some older network interface cards, loopback interfaces, Wimax cards, and some Infiniband devices, are said to be *VLAN challenged*, meaning they cannot support VLANs. This is usually because the devices cannot cope with VLAN headers and the larger MTU size associated with VLANs.

1. Create a VLAN interface file **bond0.192**:

```
~]# vi /etc/sysconfig/network-scripts/ifcfg-bond0.192
DEVICE=bond0.192
NAME=bond0.192
BOOTPROTO=none
ONPARENT=yes
IPADDR=192.168.10.1
NETMASK=255.255.255.0
VLAN=yes
NM_CONTROLLED=no
```

2. Bring up the VLAN interface as **root**:

```
~]# ifup /etc/sysconfig/network-scripts/ifcfg-bond0.192
Determining if ip address 192.168.10.1 is already in use for device bond0.192...
```

3. Enabling VLAN tagging on the network switch. Consult the documentation for the switch to see what configuration is required.
4. Check the status of the interfaces on the server:

```
~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            inet6 ::1/128 scope host
                valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    master bond0 state UP qlen 1000
    link/ether 52:54:00:19:28:fe brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    master bond0 state UP qlen 1000
    link/ether 52:54:00:f6:63:9a brd ff:ff:ff:ff:ff:ff
4: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue
    state UP
    link/ether 52:54:00:19:28:fe brd ff:ff:ff:ff:ff:ff
        inet 192.168.100.100/24 brd 192.168.100.255 scope global bond0
            inet6 fe80::5054:ff:fe19:28fe/64 scope link
                valid_lft forever preferred_lft forever
5: bond0.192@bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500
    qdisc noqueue state UP
    link/ether 52:54:00:19:28:fe brd ff:ff:ff:ff:ff:ff
```

```
inet 192.168.10.1/24 brd 192.168.10.255 scope global bond0.192
inet6 fe80::5054:ff:fe19:28fe/64 scope link
    valid_lft forever preferred_lft forever
```

Notice there is now **bond0.192@bond0** in the list of interfaces and the status is **MASTER,UP**.

- Check the route on the server:

```
~]$ ip route
192.168.100.0/24 dev bond0 proto kernel scope link src 192.168.100.100
192.168.10.0/24 dev bond0.192 proto kernel scope link src 192.168.10.1
169.254.0.0/16 dev bond0 scope link metric 1004
169.254.0.0/16 dev bond0.192 scope link metric 1005
```

Notice there is now a route for the **192.168.10.0/24** network pointing to the VLAN interface **bond0.192**.

Configuring the Second Server

Repeat the configuration steps for the second server, using different **IP** addresses but from the same subnets respectively.

Test the bond is up and the network switch is working as expected:

```
~]$ ping -c4 192.168.100.100
PING 192.168.100.100 (192.168.100.100) 56(84) bytes of data.
64 bytes from 192.168.100.100: icmp_seq=1 ttl=64 time=1.35 ms
64 bytes from 192.168.100.100: icmp_seq=2 ttl=64 time=0.214 ms
64 bytes from 192.168.100.100: icmp_seq=3 ttl=64 time=0.383 ms
64 bytes from 192.168.100.100: icmp_seq=4 ttl=64 time=0.396 ms

--- 192.168.100.100 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 0.214/0.586/1.353/0.448 ms
```

Testing the VLAN

To test that the network switch is configured for the VLAN, try to ping the first servers' VLAN interface:

```
~]# ping -c2 192.168.10.1
PING 192.168.10.1 (192.168.10.1) 56(84) bytes of data.
64 bytes from 192.168.10.1: icmp_seq=1 ttl=64 time=0.781 ms
64 bytes from 192.168.10.1: icmp_seq=2 ttl=64 time=0.977 ms
--- 192.168.10.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.781/0.879/0.977/0.098 ms
```

No packet loss suggests everything is configured correctly and that the VLAN and underlying interfaces are "up".

Optional Steps

- If required, perform further tests by removing and replacing network cables one at a time to verify that failover works as expected. Make use of the **ethtool** utility to verify which interface is connected to which cable. For example:

```
ethtool --identify ifname integer
```

Where *integer* is the number of times to flash the LED on the network interface.

- The bonding module does not support **STP**, therefore consider disabling the sending of BPDU packets from the network switch.
- If the system is not linked to the network except over the connection just configured, consider enabling the switch port to transition directly to sending and receiving. For example on a Cisco switch, by means of the **portfast** command.

11.2.6. Network Bridge

A network bridge is a Link Layer device which forwards traffic between networks based on MAC addresses and is therefore also referred to as a Layer 2 device. It makes forwarding decisions based on tables of MAC addresses which it builds by learning what hosts are connected to each network. A software bridge can be used within a Linux host in order to emulate a hardware bridge, for example in virtualization applications for sharing a NIC with one or more virtual NICs. This case will be illustrated here as an example.

To create a network bridge, create a file in the **/etc/sysconfig/network-scripts/** directory called **ifcfg-brN**, replacing *N* with the number for the interface, such as **0**.

The contents of the file is similar to whatever type of interface is getting bridged to, such as an Ethernet interface. The differences in this example are as follows:

- The **DEVICE** directive is given an interface name as its argument in the format **br*N***, where *N* is replaced with the number of the interface.
- The **TYPE** directive is given an argument **Bridge**. This directive determines the device type and the argument is case sensitive.
- The bridge interface configuration file now has the **IP** address and the physical interface has only a MAC address.
- An extra directive, **DELAY=0**, is added to prevent the bridge from waiting while it monitors traffic, learns where hosts are located, and builds a table of MAC addresses on which to base its filtering decisions. The default delay of 15 seconds is not needed if no routing loops are possible.
- The **NM_CONTROLLED=no** should be added to the Ethernet interface to prevent **NetworkManager** from altering the file. It can also be added to the bridge configuration file in case future versions of **NetworkManager** support bridge configuration.

The following is a sample bridge interface configuration file using a static **IP** address:

Example 11.4. Sample ifcfg-br0 interface configuration file

```
DEVICE=br0
TYPE=Bridge
IPADDR=192.168.1.1
NETMASK=255.255.255.0
ONBOOT=yes
BOOTPROTO=none
NM_CONTROLLED=no
DELAY=0
```

To complete the bridge another interface is created, or an existing interface is modified, and pointed to

the bridge interface. The following is a sample Ethernet interface configuration file pointing to a bridge interface. Configure your physical interface in **/etc/sysconfig/network-scripts/ifcfg-ethX**, where X is a unique number corresponding to a specific interface, as follows:

Example 11.5. Sample ifcfg-ethX interface configuration file

```
DEVICE=ethX
TYPE=Ethernet
HWADDR=AA:BB:CC:DD:EE:FF
BOOTPROTO=none
ONBOOT=yes
NM_CONTROLLED=no
BRIDGE=br0
```



NOTE

For the **DEVICE** directive, almost any interface name could be used as it does not determine the device type. Other commonly used names include **tap**, **dummy** and **bond** for example. **TYPE=Ethernet** is not strictly required. If the **TYPE** directive is not set, the device is treated as an Ethernet device (unless its name explicitly matches a different interface configuration file.)

You can see [Section 11.2, “Interface Configuration Files”](#) for a review of the directives and options used in network interface config files.



WARNING

If you are configuring bridging on a remote host, and you are connected to that host over the physical NIC you are configuring, please consider the implications of losing connectivity before proceeding. You will lose connectivity when restarting the service and may not be able to regain connectivity if any errors have been made. Console, or out-of-band access is advised.

Restart the networking service, in order for the changes to take effect, as follows:

```
service network restart
```

11.2.6.1. Network Bridge with Bond

An example of a network bridge formed from two or more bonded Ethernet interfaces will now be given as this is another common application in a virtualization environment. If you are not very familiar with the configuration files for bonded interfaces then please see [Section 11.2.4, “Channel Bonding Interfaces”](#)

Create or edit two or more Ethernet interface configuration files, which are to be bonded, as follows:

```
DEVICE=ethX
TYPE=Ethernet
```

```
USERCTL=no
SLAVE=yes
MASTER=bond0
BOOTPROTO=none
HWADDR=AA:BB:CC:DD:EE:FF
NM_CONTROLLED=no
```



NOTE

Using **ethX** as the interface name is common practice but almost any name could be used. Names such as **tap**, **dummy** and **bond** are commonly used.

Create or edit one interface configuration file, **/etc/sysconfig/network-scripts/ifcfg-bond0**, as follows:

```
DEVICE=bond0
ONBOOT=yes
BONDING_OPTS='mode=1 miimon=100'
BRIDGE=br0
NM_CONTROLLED=no
```

For further instructions and advice on configuring the bonding module and to view the list of bonding parameters, see [Section 31.8.1, “Using Channel Bonding”](#).

Create or edit one interface configuration file, **/etc/sysconfig/network-scripts/ifcfg-br0**, as follows:

```
DEVICE=br0
ONBOOT=yes
TYPE=Bridge
IPADDR=192.168.1.1
NETMASK=255.255.255.0
NM_CONTROLLED=no
```

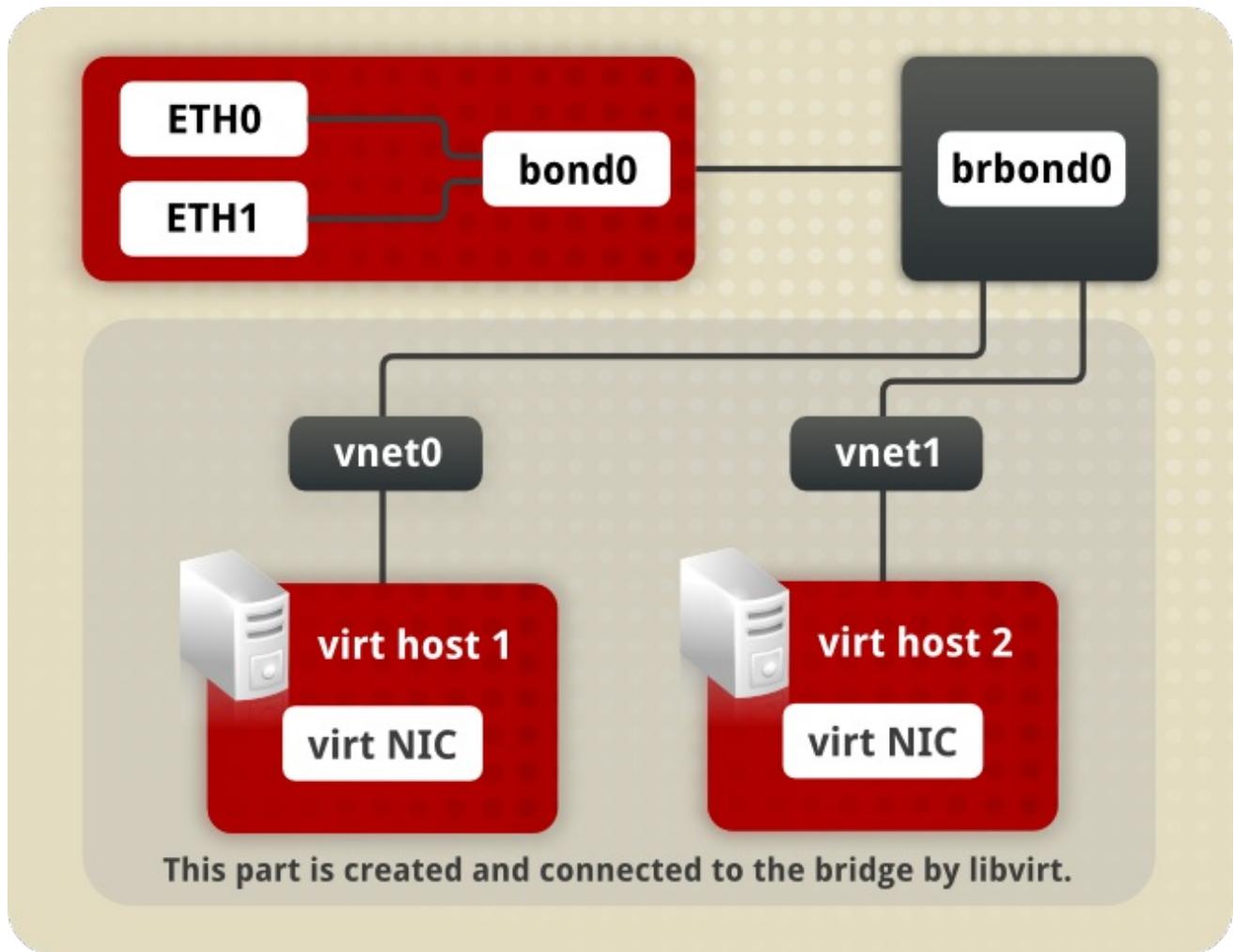


Figure 11.1. A network bridge consisting of two bonded Ethernet interfaces.

We now have two or more interface configuration files with the **MASTER=bond0** directive. These point to the configuration file named **/etc/sysconfig/network-scripts/ifcfg-bond0**, which contains the **DEVICE=bond0** directive. This **ifcfg-bond0** in turn points to the **/etc/sysconfig/network-scripts/ifcfg-br0** configuration file, which contains the **IP** address, and acts as an interface to the virtual networks inside the host.

To bring up the new or recently configured interfaces, issue a command as **root** in the following format:

```
ifup device
```

Alternatively, restart the networking service, in order for the changes to take effect, as follows:

```
~]# service network restart
```

11.2.6.2. Network Bridge with Bonded VLAN

Virtualization servers that intend to have distinct subnets for its guests while still ensuring availability in the event of a NIC failure will often combine bonds, VLANs, and bridges. An example of this configuration will now be given. By creating a bridge on the VLAN instead of the underlying device we allow VLAN tagging to be handled entirely through the host with no need to configure the guests' interfaces.

1. Ensure the bond and VLAN have been configured as outlined in [Section 11.2.5, "Configuring a VLAN over a Bond"](#).

2. Create the bridge's configuration file, **ifcfg-br0**:

```
~]# vi /etc/sysconfig/network-scripts/ifcfg-br0
DEVICE=br0
ONBOOT=yes
TYPE=Bridge
IPADDR=192.168.10.1
NETMASK=255.255.255.0
NM_CONTROLLED=no
```

3. Adjust the VLAN's configuration file, **ifcfg-bond0.192** from the earlier example, to use the newly created **br0** as its master:

```
~]# vi /etc/sysconfig/network-scripts/ifcfg-bond0.192
DEVICE=bond0.192
BOOTPROTO=none
ONPARENT=yes
#IPADDR=192.168.10.1
#NETMASK=255.255.255.0
VLAN=yes
NM_CONTROLLED=no
BRIDGE=br0
```

4. To bring up the new or recently configured interfaces, issue a command as **root** in the following format:

ifup device

Alternatively, restart the networking service, in order for the changes to take effect, as follows:

```
~]# service network restart
```

11.2.7. Setting Up 802.1Q VLAN Tagging

1. If required, start the VLAN 8021q module by issuing the following command as **root**:

```
~]# modprobe --first-time 8021q
```

No visual output indicates the module was not running and has now been loaded. Note that given a correct configuration file, the VLAN 8021q module will be loaded as required and therefore does not need to be loaded separately.

2. Configure your physical interface in **/etc/sysconfig/network-scripts/ifcfg-ethX**, where **X** is a unique number corresponding to a specific interface, as follows:

```
DEVICE=ethX
TYPE=Ethernet
BOOTPROTO=none
ONBOOT=yes
```

3. Configure the VLAN interface configuration in **/etc/sysconfig/network-scripts**. The configuration filename should be the physical interface plus a **.** character plus the VLAN ID number. For example, if the VLAN ID is 192, and the physical interface is **eth0**, then the

configuration filename should be **ifcfg-eth0.192**:

```
DEVICE=ethX.192
BOOTPROTO=none
ONBOOT=yes
IPADDR=192.168.1.1
NETMASK=255.255.255.0
USERCTL=no
NETWORK=192.168.1.0
VLAN=yes
```

If there is a need to configure a second VLAN, with for example, VLAN ID 193, on the same interface, **eth0**, add a new file with the name **eth0.193** with the VLAN configuration details.

4. Restart the networking service, in order for the changes to take effect. Issue the following command as **root**:

```
~]# service network restart
```

11.2.8. Alias and Clone Files

Two lesser-used types of interface configuration files are *alias* and *clone* files. As the **ip** utility now supports assigning multiple addresses to the same interface it is no longer necessary to use this method of binding multiple addresses to the same interface. The **ip** command to assign an address can be repeated multiple times in order to assign multiple address. For example:

```
~]# ip address add 192.168.2.223/24 dev eth1
~]# ip address add 192.168.4.223/24 dev eth1
~]# ip addr
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:fb:77:9e brd ff:ff:ff:ff:ff:ff
        inet 192.168.2.223/24 scope global eth1
            inet 192.168.4.223/24 scope global eth1
```

The commands for the **ip** utility, sometimes referred to as **iproute2** after the upstream package name, are documented in the **man ip(8)** page. The package name in Red Hat Enterprise Linux 6 is **iproute**.



NOTE

In Red Hat Enterprise Linux 6, **NetworkManager** now reads **ifcfg** alias files and assigns the addresses in them to their master interface, using the alias name as the address label. For example, if **ifcfg-eth0** and **ifcfg-eth0:1** files are present, **NetworkManager** reads the alias file's **DEVICE** line and stores this as an address label. The use of secondary addresses rather than alias is still preferred.

For new installations, users should select the **Manual** method on the **IPv4** or **IPv6** tab in **NetworkManager** to assign multiple **IP** address to the same interface. For more information on using this tool, see [Chapter 10, NetworkManager](#).

Alias interface configuration files, which are used to bind multiple addresses to a single interface, use the **ifcfg-if-name:alias-value** naming scheme.

For example, an **ifcfg-eth0:0** file could be configured to specify **DEVICE=eth0:0** and a static **IP** address

of **10.0.0.2**, serving as an alias of an Ethernet interface already configured to receive its **IP** information via **DHCP** in **ifcfg-eth0**. Under this configuration, **eth0** is bound to a dynamic **IP** address, but the same physical network card can receive requests via the fixed, **10.0.0.2 IP** address.



WARNING

Alias interfaces do not support **DHCP**.

A clone interface configuration file should use the following naming convention: **ifcfg-if-name-clone-name**. While an alias file allows multiple addresses for an existing interface, a clone file is used to specify additional options for an interface. For example, a standard **DHCP** Ethernet interface called **eth0**, may look similar to this:

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
```

Since the default value for the **USERCTL** directive is **no** if it is not specified, users cannot bring this interface up and down. To give users the ability to control the interface, create a clone by copying **ifcfg-eth0** to **ifcfg-eth0-user** and add the following line to **ifcfg-eth0-user**:

```
USERCTL=yes
```

This way a user can bring up the **eth0** interface using the **/sbin/ifup eth0-user** command because the configuration options from **ifcfg-eth0** and **ifcfg-eth0-user** are combined. While this is a very basic example, this method can be used with a variety of options and interfaces.

It is no longer possible to create alias and clone interface configuration files using a graphical tool. However, as explained at the beginning of this section, it is no longer necessary to use this method as it is now possible to directly assign multiple **IP** address to the same interface. For new installations, users should select the **Manual** method on the **IPv4** or **IPv6** tab in **NetworkManager** to assign multiple **IP** address to the same interface. For more information on using this tool, see [Chapter 10, NetworkManager](#).

11.2.9. Dialup Interfaces

If you are connecting to the Internet via a dialup connection, a configuration file is necessary for the interface.

PPP interface files are named using the following format:

ifcfg-pppX

where **X** is a unique number corresponding to a specific interface.

The **PPP** interface configuration file is created automatically when **wvdial**, or **Kppp** is used to create a dialup account. It is also possible to create and edit this file manually.

The following is a typical **/etc/sysconfig/network-scripts/ifcfg-ppp0** file:

```
DEVICE=ppp0
NAME=test
WVDIALSECT=test
MODEMPORT=/dev/modem
LINESPEED=115200
PAPNAME=test
USERCTL=true
ONBOOT=no
PERSIST=no
DEFROUTE=yes
PEERDNS=yes
DEMAND=no
IDLETIMEOUT=600
```

Serial Line Internet Protocol (SLIP) is another dialup interface, although it is used less frequently. **SLIP** files have interface configuration file names such as **ifcfg-slo**.

Other options that may be used in these files include:

DEFROUTE=answer

where *answer* is one of the following:

- **yes** – Set this interface as the default route.
- **no** – Do not set this interface as the default route.

DEMAND=answer

where *answer* is one of the following:

- **yes** – This interface allows **pppd** to initiate a connection when someone attempts to use it.
- **no** – A connection must be manually established for this interface.

IDLETIMEOUT=value

where *value* is the number of seconds of idle activity before the interface disconnects itself.

INITSTRING=string

where *string* is the initialization string passed to the modem device. This option is primarily used in conjunction with **SLIP** interfaces.

LINESPEED=value

where *value* is the baud rate of the device. Possible standard values include **57600**, **38400**, **19200**, and **9600**.

MODEMPORT=device

where *device* is the name of the serial device that is used to establish the connection for the interface.

MTU=value

where *value* is the *Maximum Transfer Unit* (MTU) setting for the interface. The MTU refers to the largest number of bytes of data a frame can carry, not counting its header information. In some dialup situations, setting this to a value of **576** results in fewer packets dropped and a slight

improvement to the throughput for a connection.

NAME=name

where *name* is the reference to the title given to a collection of dialup connection configurations.

PAPNAME=name

where *name* is the user name given during the *Password Authentication Protocol* (PAP) exchange that occurs to allow connections to a remote system.

PERSIST=answer

where *answer* is one of the following:

- **yes** – This interface should be kept active at all times, even if deactivated after a modem hangs up.
- **no** – This interface should not be kept active at all times.

REMIP=address

where *address* is the **IP** address of the remote system. This is usually left unspecified.

WVDIALSELECT=name

where *name* associates this interface with a dialer configuration in **/etc/wvdial.conf**. This file contains the phone number to be dialed and other important information for the interface.

11.2.10. Other Interfaces

Other common interface configuration files include the following:

ifcfg-lo

A local *loopback interface* is often used in testing, as well as being used in a variety of applications that require an **IP** address pointing back to the same system. Any data sent to the loopback device is immediately returned to the host's network layer.



WARNING

The loopback interface script, **/etc/sysconfig/network-scripts/ifcfg-lo**, should never be edited manually. Doing so can prevent the system from operating correctly.

ifcfg-irlan0

An *infrared interface* allows information between devices, such as a laptop and a printer, to flow over an infrared link. This works in a similar way to an Ethernet device except that it commonly occurs over a peer-to-peer connection.

ifcfg-plip0

A *Parallel Line Interface Protocol* (PLIP) connection works much the same way as an Ethernet device, except that it utilizes a parallel port.

Interface configuration files for Linux on System z include the following:

ifcfg-hsiN

A *HiperSockets* interface is an interface for high-speed TCP/IP communication within and across z/VM guest virtual machines and logical partitions (LPARs) on an IBM System z mainframe.

11.3. INTERFACE CONTROL SCRIPTS

The interface control scripts activate and deactivate system interfaces. There are two primary interface control scripts that call on control scripts located in the **/etc/sysconfig/network-scripts/** directory: **/sbin/ifdown** and **/sbin/ifup**.

The **ifup** and **ifdown** interface scripts are symbolic links to scripts in the **/sbin/** directory. When either of these scripts are called, they require the value of the interface to be specified, such as:

```
ifup eth0
```

WARNING



The **ifup** and **ifdown** interface scripts are the only scripts that the user should use to bring up and take down network interfaces.

The following scripts are described for reference purposes only.

Two files used to perform a variety of network initialization tasks during the process of bringing up a network interface are **/etc/rc.d/init.d/functions** and **/etc/sysconfig/network-scripts/network-functions**. See [Section 11.7, "Network Function Files"](#) for more information.

After verifying that an interface has been specified and that the user executing the request is allowed to control the interface, the correct script brings the interface up or down. The following are common interface control scripts found within the **/etc/sysconfig/network-scripts/** directory:

ifup-aliases

Configures **IP** aliases from interface configuration files when more than one **IP** address is associated with an interface.

ifup-ppp and ifdown-ppp

Brings ISDN interfaces up and down.

ifup-ipv6 and ifdown-ipv6

Brings **IPv6** interfaces up and down.

ifup-plip

Brings up a **PLIP** interface.

ifup-plusb

Brings up a USB interface for network connections.

ifup-post and **ifdown-post**

Contains commands to be executed after an interface is brought up or down.

ifup-ppp and **ifdown-ppp**

Brings a **PPP** interface up or down.

ifup-routes

Adds static routes for a device as its interface is brought up.

ifdown-sit and **ifup-sit**

Contains function calls related to bringing up and down an **IPv6** tunnel within an **IPv4** connection.

ifup-wireless

Brings up a wireless interface.



WARNING

Removing or modifying any scripts in the **/etc/sysconfig/network-scripts/** directory can cause interface connections to act irregularly or fail. Only advanced users should modify scripts related to a network interface.

The easiest way to manipulate all network scripts simultaneously is to use the **/sbin/service** command on the network service (**/etc/rc.d/init.d/network**), as illustrated by the following command:

/sbin/service network action

Here, *action* can be either **start**, **stop**, or **restart**.

To view a list of configured devices and currently active network interfaces, use the following command:

/sbin/service network status

11.4. STATIC ROUTES AND THE DEFAULT GATEWAY

Static routes are for traffic that must not, or should not, go through the default gateway. Routing is often handled by devices on the network dedicated to routing (although any device can be configured to perform routing). Therefore, it is often not necessary to configure static routes on Red Hat Enterprise Linux servers or clients. Exceptions include traffic that must pass through an encrypted VPN

tunnel or traffic that should take a specific route for reasons of cost or security. The default gateway is for any and all traffic which is not destined for the local network and for which no preferred route is specified in the routing table. The default gateway is traditionally a dedicated network router.

Configuring Static Routes Using the Command Line

If static routes are required, they can be added to the routing table by means of the **ip route add** command and removed using the **ip route del** command. The more frequently used **ip route** commands take the following form:

```
ip route [ add | del | change | append | replace ] destination-address
```

See the **ip-route(8)** man page for more details on the options and formats.

Use the **ip route** command without options to display the **IP** routing table. For example:

```
~]$ ip route
default via 192.168.122.1 dev eth0 proto static metric 1024
192.168.122.0/24 dev ens9 proto kernel scope link src 192.168.122.107
192.168.122.0/24 dev eth0 proto kernel scope link src 192.168.122.126
```

To add a static route to a host address, in other words to a single **IP** address, issue a command as **root**:

```
~]# ip route add 192.0.2.1 via 10.0.0.1 [dev ifname]
```

Where **192.0.2.1** is the **IP** address of the host in dotted decimal notation, **10.0.0.1** is the next hop address and **ifname** is the exit interface leading to the next hop.

To add a static route to a network, in other words to an **IP** address representing a range of **IP** addresses, issue the following command as **root**:

```
~]# ip route add 192.0.2.0/24 via 10.0.0.1 [dev ifname]
```

where **192.0.2.0** is the **IP** address of the destination network in dotted decimal notation and **/24** is the network prefix. The network prefix is the number of enabled bits in the subnet mask. This format of network address slash network prefix length is sometimes referred to as *classless inter-domain routing* (CIDR) notation.

Static route configuration can be stored per-interface in a **/etc/sysconfig/network-scripts/route-*interface*** file. For example, static routes for the **eth0** interface would be stored in the **/etc/sysconfig/network-scripts/route-eth0** file. The **route-interface** file has two formats: **ip** command arguments and network/netmask directives. These are described below.

See the **ip-route(8)** man page for more information on the **ip route** command.

Configuring The Default Gateway

The default gateway is determined by the network scripts which parse the **/etc/sysconfig/network** file first and then the network interface **ifcfg** files for interfaces that are “up”. The **ifcfg** files are parsed in numerically ascending order, and the last GATEWAY directive to be read is used to compose a default route in the routing table.

The default route can thus be indicated by means of the GATEWAY directive and can be specified either globally or in interface-specific configuration files. Specifying the gateway globally has certain advantages in static networking environments, especially if more than one network interface is present. It can make fault finding simpler if applied consistently. There is also the GATEWAYDEV directive, which

is a global option. If multiple devices specify GATEWAY, and one interface uses the GATEWAYDEV directive, that directive will take precedence. This option is not recommended as it can have unexpected consequences if an interface goes down and it can complicate fault finding.

In dynamic network environments, where mobile hosts are managed by **NetworkManager**, gateway information is likely to be interface specific and is best left to be assigned by **DHCP**. In special cases where it is necessary to influence **NetworkManager**'s selection of the exit interface to be used to reach a gateway, make use of the **DEFROUTE=no** command in the **ifcfg** files for those interfaces which do not lead to the default gateway.

Global default gateway configuration is stored in the **/etc/sysconfig/network** file. This file specifies gateway and host information for all network interfaces. For more information about this file and the directives it accepts, see [Section D.1.14, "/etc/sysconfig/network"](#).

11.5. CONFIGURING STATIC ROUTES IN IFCFG FILES

Static routes set using **ip** commands at the command prompt will be lost if the system is shutdown or restarted. To configure static routes to be persistent after a system restart, they must be placed in per-interface configuration files in the **/etc/sysconfig/network-scripts/** directory. The file name should be of the format **route-ifname**. There are two types of commands to use in the configuration files; **ip** commands as explained in [Section 11.5.1, "Static Routes Using the IP Command Arguments Format"](#) and the **Network/Netmask** format as explained in [Section 11.5.2, "Network/Netmask Directives Format"](#).

11.5.1. Static Routes Using the IP Command Arguments Format

If required in a per-interface configuration file, for example **/etc/sysconfig/network-scripts/route-eth0**, define a route to a default gateway on the first line. This is only required if the gateway is not set via **DHCP** and is not set globally in the **/etc/sysconfig/network** file:

```
default via 192.168.1.1 dev interface
```

where **192.168.1.1** is the **IP** address of the default gateway. The **interface** is the interface that is connected to, or can reach, the default gateway. The **dev** option can be omitted, it is optional. Note that this setting takes precedence over a setting in the **/etc/sysconfig/network** file.

If a route to a remote network is required, a static route can be specified as follows. Each line is parsed as an individual route:

```
10.10.10.0/24 via 192.168.1.1 [dev interface]
```

where **10.10.10.0/24** is the network address and prefix length of the remote or destination network. The address **192.168.1.1** is the **IP** address leading to the remote network. It is preferably the **next hop address** but the address of the exit interface will work. The “next hop” means the remote end of a link, for example a gateway or router. The **dev** option can be used to specify the exit interface **interface** but it is not required. Add as many static routes as required.

The following is an example of a **route-interface** file using the **ip** command arguments format. The default gateway is **192.168.0.1**, interface eth0 and a leased line or WAN connection is available at **192.168.0.10**. The two static routes are for reaching the **10.10.10.0/24** network and the **172.16.1.10/32** host:

```
default via 192.168.0.1 dev eth0
10.10.10.0/24 via 192.168.0.10 dev eth0
172.16.1.10/32 via 192.168.0.10 dev eth0
```

In the above example, packets going to the local **192.168.0.0/24** network will be directed out the interface attached to that network. Packets going to the **10.10.10.0/24** network and **172.16.1.10/32** host will be directed to **192.168.0.10**. Packets to unknown, remote, networks will use the default gateway therefore static routes should only be configured for remote networks or hosts if the default route is not suitable. Remote in this context means any networks or hosts that are not directly attached to the system.

Specifying an exit interface is optional. It can be useful if you want to force traffic out of a specific interface. For example, in the case of a VPN, you can force traffic to a remote network to pass through a tun0 interface even when the interface is in a different subnet to the destination network.



IMPORTANT

If the default gateway is already assigned from **DHCP**, the **IP** command arguments format can cause one of two errors during start-up, or when bringing up an interface from the down state using the **ifup** command: "RTNETLINK answers: File exists" or 'Error: either "to" is a duplicate, or "X.X.X.X" is a garbage.', where X.X.X.X is the gateway, or a different **IP** address. These errors can also occur if you have another route to another network using the default gateway. Both of these errors are safe to ignore.

11.5.2. Network/Netmask Directives Format

You can also use the network/netmask directives format for **route-interface** files. The following is a template for the network/netmask format, with instructions following afterwards:

```
ADDRESS0=10.10.10.0
NETMASK0=255.255.255.0
GATEWAY0=192.168.1.1
```

- **ADDRESS0=10.10.10.0** is the network address of the remote network or host to be reached.
- **NETMASK0=255.255.255.0** is the netmask for the network address defined with **ADDRESS0=10.10.10.0**.
- **GATEWAY0=192.168.1.1** is the default gateway, or an **IP** address that can be used to reach **ADDRESS0=10.10.10.0**

The following is an example of a **route-interface** file using the network/netmask directives format. The default gateway is **192.168.0.1** but a leased line or WAN connection is available at **192.168.0.10**. The two static routes are for reaching the **10.10.10.0/24** and **172.16.1.0/24** networks:

```
ADDRESS0=10.10.10.0
NETMASK0=255.255.255.0
GATEWAY0=192.168.0.10
ADDRESS1=172.16.1.10
NETMASK1=255.255.255.0
GATEWAY1=192.168.0.10
```

Subsequent static routes must be numbered sequentially, and must not skip any values. For example, **ADDRESS0**, **ADDRESS1**, **ADDRESS2**, and so on.

11.6. CONFIGURING IPV6 TOKENIZED INTERFACE IDENTIFIERS

In a network, servers are generally given static addresses and these are usually configured manually to

avoid relying on a **DHCP** server which may fail or run out of addresses. The **IPv6** protocol introduced *Stateless Address Autoconfiguration* (SLAAC) which enables clients to assign themselves an address without relying on a **DHCPv6** server. SLLAC derives the **IPv6** address based on the interface hardware, therefore it should not be used for servers in case the hardware is changed and the associated SLLAC generated address changes with it. In an **IPv6** environment, if the network prefix is changed, or the system is moved to a new location, any manually configured static addresses would have to be edited due to the changed prefix.

To address these problems, the IETF draft *Tokenised IPv6 Identifiers* has been implemented in the kernel together with corresponding additions to the **ip** utility. This enables the lower 64 bit interface identifier part of the **IPv6** address to be based on a token, supplied by the administrator, leaving the network prefix, the higher 64 bits, to be obtained from *router advertisements* (RA). This means that if the network interface hardware is changed, the lower 64 bits of the address will not change, and if the system is moved to another network, the network prefix will be obtained from router advertisements automatically, thus no manual editing is required.

To configure an interface to use a tokenized **IPv6** identifier, issue a command in the following format as **root** user:

```
~]# ip token set ::1a:2b:3c:4d/64 dev eth4
```

Where **::1a:2b:3c:4d/64** is the token to be used. This setting is not persistent. To make it persistent, add the command to an init script. See [Section 11.3, “Interface Control Scripts”](#).

Using a memorable token is possible, but is limited to the range of valid hexadecimal digits. For example, for a **DNS** server, which traditionally uses port **53**, a token of **::53/64** could be used.

To view all the configured **IPv6** tokens, issue the following command:

```
~]$ ip token
    token :: dev eth0
    token :: dev eth1
    token :: dev eth2
    token :: dev eth3
    token ::1a:2b:3c:4d dev eth4
```

To view the configured **IPv6** token for a specific interface, issue the following command:

```
~]$ ip token get dev eth4
    token ::1a:2b:3c:4d dev eth4
```

Note that adding a token to an interface will replace a previously allocated token, and in turn invalidate the address derived from it. Supplying a new token causes a new address to be generated and applied, but this process will leave any other addresses unchanged. In other words, a new tokenized identifier only replaces a previously existing tokenized identifier, not any other **IP** address.



NOTE

Take care not to add the same token to more than one system or interface as the duplicate address detection (DAD) mechanism will not be able to resolve the problem. Once a token is set, it cannot be cleared or reset, except by rebooting the machine.

11.7. NETWORK FUNCTION FILES

Red Hat Enterprise Linux makes use of several files that contain important common functions used to bring interfaces up and down. Rather than forcing each interface control file to contain these functions, they are grouped together in a few files that are called upon when necessary.

The **/etc/sysconfig/network-scripts/network-functions** file contains the most commonly used **IPv4** functions, which are useful to many interface control scripts. These functions include contacting running programs that have requested information about changes in the status of an interface, setting host names, finding a gateway device, verifying whether or not a particular device is down, and adding a default route.

As the functions required for **IPv6** interfaces are different from **IPv4** interfaces, a **/etc/sysconfig/network-scripts/network-functions-ipv6** file exists specifically to hold this information. The functions in this file configure and delete static **IPv6** routes, create and remove tunnels, add and remove **IPv6** addresses to an interface, and test for the existence of an **IPv6** address on an interface.

11.8. ETHTOOL

Ethtool is a utility for configuration of *Network Interface Cards* (NICs). This utility allows querying and changing settings such as speed, port, auto-negotiation, PCI locations and checksum offload on many network devices, especially Ethernet devices.

We present here a short selection of often used **ethtool** commands together with some useful commands that are not well known. For a full list of commands type **ethtool -h** or see the man page, **ethtool(8)**, for a more comprehensive list and explanation. The first two examples are information queries and show the use of the different formats of the command.

But first, the command structure:

ethtool [*option...*] *devname*

where *option* is none or more options, and *devname* is your Network Interface Card (NIC). For example *eth0* or *em1*.

ethtool

The **ethtool** command with only a device name as an option is used to print the current settings of the specified device. It takes the following form:

ethtool *devname*

where *devname* is your NIC. For example *eth0* or *em1*.

Some values can only be obtained when the command is run as **root**. Here is an example of the output when the command is run as **root**:

```
~]# ethtool em1
Settings for em1:
Supported ports: [ TP ]
Supported link modes: 10baseT/Half 10baseT/Full
                      100baseT/Half 100baseT/Full
                      1000baseT/Full
Supported pause frame use: No
Supports auto-negotiation: Yes
Advertised link modes: 10baseT/Half 10baseT/Full
                      100baseT/Half 100baseT/Full
```

```

1000baseT/Full
Advertised pause frame use: No
Advertised auto-negotiation: Yes
Speed: 1000Mb/s
Duplex: Full
Port: Twisted Pair
PHYAD: 2
Transceiver: internal
Auto-negotiation: on
MDI-X: on
Supports Wake-on: pumbg
Wake-on: g
Current message level: 0x00000007 (7)
    drv probe link
Link detected: yes

```

Issue the following command, using the short or long form of the argument, to query the specified network device for associated driver information:

```
ethtool -i, --driver devname
```

where *devname* is your Network Interface Card (NIC). For example `eth0` or `em1`.

Here is an example of the output:

```

~]$ ethtool -i em1
driver: e1000e
version: 2.0.0-k
firmware-version: 0.13-3
bus-info: 0000:00:19.0
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes

```

Here follows a list of command options to query, identify or reset the device. They are in the usual **-short** and **--long** form:

--statistics

The **--statistics** or **-S** queries the specified network device for NIC and driver statistics. It takes the following form:

```
-S, --statistics devname
```

where *devname* is your NIC.

--identify

The **--identify** or **-p** option initiates adapter-specific action intended to enable an operator to easily identify the adapter by sight. Typically this involves blinking one or more LEDs on the specified network port. It takes the following form:

```
-p, --identify devname integer
```

where *integer* is length of time in seconds to perform the action,

and *devname* is your NIC.

--show-time-stamping

The **--show-time-stamping** or **-T** option queries the specified network device for time stamping parameters. It takes the following form:

-T, --show-time-stamping *devname*

where *devname* is your NIC.

--show-offload

The **--show-features**, or **--show-offload**, or **-k** option queries the specified network device for the state of protocol offload and other features. It takes the following form:

-k, --show-features, --show-offload *devname*

where *devname* is your NIC.

--test

The **--test** or **-t** option is used to perform tests on a Network Interface Card. It takes the following form:

-t, --test *devname word*

where *word* is one of the following:

- **offline** – Perform a comprehensive set of tests. Service will be interrupted.
- **online** – Perform a reduced set of tests. Service should not be interrupted.
- **external_lb** – Perform full set of tests including loopback tests while fitted with a loopback cable.

and *devname* is your NIC.

Changing some or all settings of the specified network device requires the **-s** or **--change** option. All the following options are only applied if the **-s** or **--change** option is also specified. For the sake of clarity we will omit it here.

To make these settings permanent you can make use of the **ETHTOOL_OPTS** directive. It can be used in interface configuration files to set the desired options when the network interface is brought up. See [Section 11.2.1, “Ethernet Interfaces”](#) for more details on how to use this directive.

--offload

The **--features**, or **--offload**, or **-K** option changes the offload parameters and other features of the specified network device. It takes the following form:

-K, --features, --offload *devname feature boolean*

where *feature* is a built-in or kernel supplied feature,

boolean is one of **ON** or **OFF**,

and *devname* is your NIC.

The **ethtool(8)** man page lists most features. As the feature set is dependent on the NIC driver, you should consult the driver documentation for features not listed in the man page.

--speed

The **--speed** option is used to set the speed in megabits per second (Mb/s). Omitting the speed value will show the supported device speeds. It takes the following form:

--speed *number devname*

where *number* is the speed in megabits per second (Mb/s),

and *devname* is your NIC.

--duplex

The **--duplex** option is used to set the transmit and receive mode of operation. It takes the following form:

--duplex *word devname*

where *word* is one of the following:

- **half** – Sets half-duplex mode. Usually used when connected to a hub.
- **full** – Sets full-duplex mode. Usually used when connected to a switch or another host.

and *devname* is your NIC.

--port

The **--port** option is used to select the device port . It takes the following form:

--port *value devname*

where *value* is one of the following:

- **tp** – An Ethernet interface using Twisted-Pair cable as the medium.
- **aui** – Attachment Unit Interface (AUI). Normally used with hubs.
- **bnc** – An Ethernet interface using BNC connectors and co-axial cable.
- **mii** – An Ethernet interface using a Media Independent Interface (MII).
- **fibre** – An Ethernet interface using Optical Fibre as the medium.

and *devname* is your NIC.

--autoneg

The **--autoneg** option is used to control auto-negotiation of network speed and mode of operation (full-duplex or half-duplex mode). If auto-negotiation is enabled you can initiate re-negotiation of

network speeds and mode of operation by using the **-r, --negotiate** option. You can display the auto-negotiation state using the **--a, --show-pause** option.

It takes the following form:

```
--autoneg value devname
```

where *value* is one of the following:

- **yes** – Allow auto-negotiating of network speed and mode of operation.
- **no** – Do not allow auto-negotiating of network speed and mode of operation.

and *devname* is your NIC.

--advertise

The **--advertise** option is used to set what speeds and modes of operation (duplex mode) are advertised for auto-negotiation. The argument is one or more hexadecimal values from [Table 11.1, "Ehtool advertise options: speed and mode of operation"](#).

It takes the following form:

```
--advertise option devname
```

where *option* is one or more of the hexadecimal values from the table below and *devname* is your NIC.

Table 11.1. Ehtool advertise options: speed and mode of operation

Hex Value	Speed	Duplex Mode	IEEE standard?
0x001	10	Half	Yes
0x002	10	Full	Yes
0x004	100	Half	Yes
0x008	100	Full	Yes
0x010	1000	Half	No
0x020	1000	Full	Yes
0x8000	2500	Full	Yes
0x1000	10000	Full	Yes
0x20000	20000MLD2	Full	No
0x20000	20000MLD2	Full	No

Hex Value	Speed	Duplex Mode	IEEE standard?
0x40000	20000KR2	Full	No

--phyad

The **--phyad** option is used to change the physical address. Often referred to as the MAC or hardware address but in this context referred to as the physical address.

It takes the following form:

```
--phyad physical_address devname
```

where *physical_address* is the physical address in hexadecimal format and *devname* is your NIC.

--xcvr

The **--xcvr** option is used to select the transceiver type. Currently only "internal" and "external" can be specified. In the future other types might be added.

It takes the following form:

```
--xcvr word devname
```

where *word* is one of the following:

- **internal** – Use internal transceiver.
- **external** – Use external transceiver.

and *devname* is your NIC.

--wol

The **--wol** option is used to set "Wake-on-LAN" options. Not all devices support this. The argument to this option is a string of characters specifying which options to enable.

It takes the following form:

```
--wol value devname
```

where *value* is one or more of the following:

- **p** – Wake on PHY activity.
- **u** – Wake on unicast messages.
- **m** – Wake on multicast messages.
- **b** – Wake on broadcast messages.
- **g** – Wake-on-Lan; wake on receipt of a "magic packet".
- **s** – Enable security function using password for Wake-on-Lan.

- **d** – Disable Wake-on-Lan and clear all settings.

and *devname* is your NIC.

--sopass

The **--sopass** option is used to set the “SecureOn” password. The argument to this option must be 6 bytes in Ethernet MAC hexadecimal format (xx:yy:zz:aa:bb:cc).

It takes the following form:

```
--sopass xx:yy:zz:aa:bb:cc devname
```

where *xx:yy:zz:aa:bb:cc* is the password in the same format as a MAC address and *devname* is your NIC.

--msglvl

The **--msglvl** option is used to set the driver message-type flags by name or number. The precise meanings of these type flags differ between drivers.

It takes the following form:

```
--msglvl message_type devname
```

where *message_type* is one of:

- message type name in plain text.
- hexadecimal number indicating the message type.

and *devname* is your NIC.

The defined message type names and numbers are shown in the table below:

Table 11.2. Driver message type

Message Type	Hex Value	Description
drv	0x0001	General driver status
probe	0x0002	Hardware probing
link	0x0004	Link state
timer	0x0008	Periodic status check
ifdown	0x0010	Interface being brought down
ifup	0x0020	Interface being brought up
rx_err	0x0040	Receive error

Message Type	Hex Value	Description
tx_err	0x0080	Transmit error
intr	0x0200	Interrupt handling
tx_done	0x0400	Transmit completion
rx_status	0x0800	Receive completion
pktdata	0x1000	Packet contents
hw	0x2000	Hardware status
wol	0x4000	Wake-on-LAN status

11.9. CONFIGURING NETCONSOLE

The **netconsole** kernel module enables logging of kernel messages over the network to another computer. It allows kernel debugging when disk logging fails or when using the serial console is not possible.

Configuring a Listening Machine

To enable receiving **netconsole** logging messages, install the **rsyslog** package:

```
# yum install rsyslog
```

To configure **rsyslogd** to listen on the 514/UDP port and receive messages from the network, uncomment the following lines in the **MODULES** section of **/etc/rsyslog.conf**:

```
$ModLoad imudp
$UDPServerRun 514
```

Restart the **rsyslogd** service for the changes to take effect:

```
# service rsyslog restart
```

To verify that **rsyslogd** is listening on the 514/udp port, use the following command:

```
# netstat -l | grep syslog
udp    0    0 *:syslog          *:*
udp    0    0 *:syslog          *:*
```

The **0 *:syslog** value in the **netstat -l** output mean that **rsyslogd** is listening on default **netconsole** port, which is defined in the **/etc/services** file:

```
$ cat /etc/services | grep syslog
syslog      514/udp
```

```

syslog-conn 601/tcp      # Reliable Syslog Service
syslog-conn 601/udp      # Reliable Syslog Service
syslog-tls   6514/tcp    # Syslog over TLS

```

Configuring a Sending Machine

In Red Hat Enterprise Linux 6, **netconsole** is configured using the file **/etc/sysconfig/netconsole**, which is part of the **initscripts** package. This package is installed by default and it also provides the **netconsole** service.

To configure a sending machine, set the value of the **SYSLOGADDR** variable in the **/etc/sysconfig/netconsole** file to match the IP address of the **syslogd** server, for example:

```
SYSLOGADDR=192.168.0.1
```

Restart the **netconsole** service so the changes take effect. Then, use the **chkconfig** command to ensure **netconsole** service starts automatically after next reboot:

```

]# service netconsole restart
Initializing netconsole                                [ OK ]
]# chkconfig netconsole on

```

By default, the **rsyslogd** server writes the **netconsole** messages from the client in **/var/log/messages** or in the file specified in **rsyslog.conf**.



NOTE

To set **rsyslogd** and **netconsole** to use a different port, change the following line in **/etc/rsyslog.conf** to the desired port number:

```
$UDPServerRun <PORT>
```

On the sending machine, uncomment and edit the following line in the **/etc/sysconfig/netconsole** file:

```
SYSLOGPORT=514
```

For more information about **netconsole** configuration and troubleshooting tips, see [Netconsole Kernel Documentation](#).

11.10. ADDITIONAL RESOURCES

The following are resources which explain more about network interfaces.

Installed Documentation

- **/usr/share/doc/initscripts-version/sysconfig.txt** – A guide to available options for network configuration files, including **IPv6** options not covered in this chapter.

Online Resources

- <http://linux-ip.net/gi/ip-cref/> – This document contains a wealth of information about the **ip** command, which can be used to manipulate routing tables, among other things.

- [Red Hat Access Labs](#) – The Red Hat Access Labs includes a “Network Bonding Helper”.

See Also

- [Appendix E, *The proc File System*](#) – Describes the **sysctl** utility and the virtual files within the **/proc** directory, which contain networking parameters and statistics among other things.

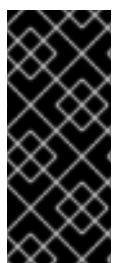
PART V. INFRASTRUCTURE SERVICES

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

CHAPTER 12. 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 explains the concept of runlevels, and describes how to set the default one. It also covers the setup of the services to be run in each of these runlevels, and provides information on how to start, stop, and restart the services on the command line using the **service** command.



IMPORTANT

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, see the Red Hat Enterprise Linux 6 Security Guide.

12.1. CONFIGURING THE DEFAULT RUNLEVEL

A *runlevel* is a state, or *mode*, defined by services that are meant to be run when this runlevel is selected. Seven numbered runlevels exist (indexed from 0):

Table 12.1. Runlevels in Red Hat Enterprise Linux

Runlevel	Description
0	Used to halt the system. This runlevel is reserved and cannot be changed.
1	Used to run in a single-user mode. This runlevel is reserved and cannot be changed.
2	Not used by default. You are free to define it yourself.
3	Used to run in a full multi-user mode with a command-line user interface.
4	Not used by default. You are free to define it yourself.
5	Used to run in a full multi-user mode with a graphical user interface.
6	Used to reboot the system. This runlevel is reserved and cannot be changed.

To check in which runlevel you are operating, type the following:

```
~]$ runlevel
N 5
```

The **runlevel** command displays previous and current runlevel. In this case it is number 5, which means the system is running in a full multi-user mode with a graphical user interface.

The default runlevel can be changed by modifying the **/etc/inittab** file, which contains a line near the end of the file similar to the following:

```
| id:5:initdefault:
```

To do so, edit this file as **root** and change the number on this line to the desired value. The change will take effect the next time you reboot the system.

12.2. CONFIGURING THE SERVICES

To allow you to configure which services are started at boot time, Red Hat Enterprise Linux is shipped with the following utilities: the **Service Configuration** graphical application, the **ntsysv** text user interface, and the **chkconfig** command-line tool.



IMPORTANT

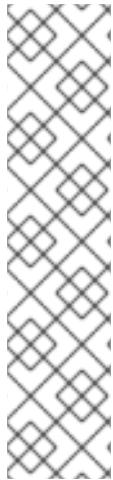
To ensure optimal performance on POWER architecture, it is recommended that the **irqbalance** service is enabled. In most cases, this service is installed and configured to run during the Red Hat Enterprise Linux 6 installation. To verify that **irqbalance** is running, as **root**, type the following at a shell prompt:

```
| ~]# service irqbalance status  
irqbalance (pid 1234) is running...
```

For information on how to enable and run a service using a graphical user interface, see [Section 12.2.1, “Using the Service Configuration Utility”](#). For instructions on how to perform these task on the command line, see [Section 12.2.3, “Using the chkconfig Utility”](#) and [Section 12.3, “Running Services”](#) respectively.

12.2.1. Using the Service Configuration Utility

The **Service Configuration** utility is a graphical application developed by Red Hat to configure which services are started in a particular runlevel, as well as to start, stop, and restart them from the menu. To start the utility, select **System → Administration → Services** from the panel, or type the command **system-config-services** at a shell prompt.



NOTE

The **system-config-services** utility is provided by the **system-config-services** package, which may not be installed by default on your version of Red Hat Enterprise Linux. To ensure that, first run the following command:

```
| ~]$ rpm -q system-config-services
```

If the package is not installed by default, install it manually by running the following command as root:

```
| ~]# yum install system-config-services
```

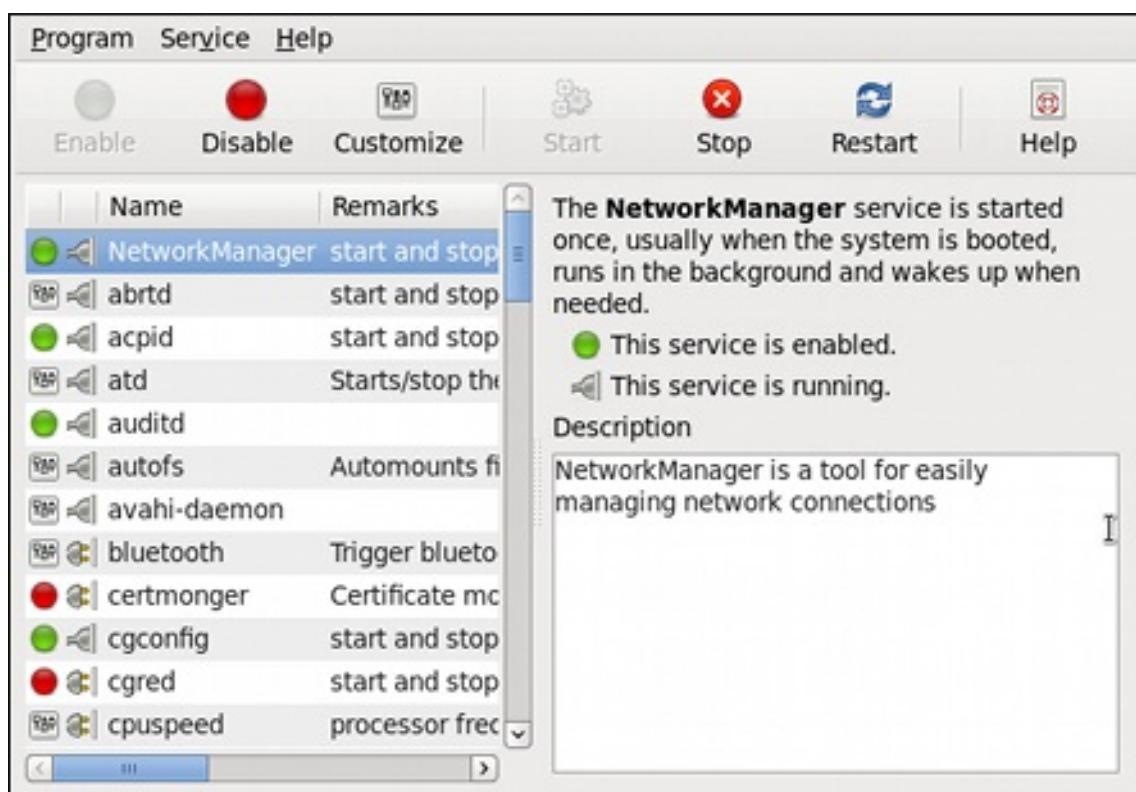


Figure 12.1. The Service Configuration utility

The utility displays the list of all available services (services from the `/etc/rc.d/init.d` directory, as well as services controlled by `xinetd`) along with their description and the current status. For a complete list of used icons and an explanation of their meaning, see [Table 12.2, “Possible service states”](#).

Note that unless you are already authenticated, you will be prompted to enter the superuser password the first time you make a change.

Table 12.2. Possible service states

Icon	Description
	The service is enabled.
	The service is disabled.
	The service is enabled for selected runlevels only.
	The service is running.
	The service is stopped.
	There is something wrong with the service.
	The status of the service is unknown.

Icon	Description
	Enables a service.

12.2.1.1. Enabling and Disabling a Service

To enable a service, select it from the list and either click the **Enable** button on the toolbar, or choose **Service → Enable** from the main menu.

To disable a service, select it from the list and either click the **Disable** button on the toolbar, or choose **Service → Disable** from the main menu.

12.2.1.2. Starting, Restarting, and Stopping a Service

To start a service, select it from the list and either click the **Start** button on the toolbar, or choose **Service → Start** from the main menu. Note that this option is not available for services controlled by **xinetd**, as they are started by it on demand.

To restart a running service, select it from the list and either click the **Restart** button on the toolbar, or choose **Service → Restart** from the main menu. Note that this option is not available for services controlled by **xinetd**, as they are started and stopped by it automatically.

To stop a service, select it from the list and either click the **Stop** button on the toolbar, or choose **Service → Stop** from the main menu. Note that this option is not available for services controlled by **xinetd**, as they are stopped by it when their job is finished.

12.2.1.3. Selecting Runlevels

To enable the service for certain runlevels only, select it from the list and either click the **Customize** button on the toolbar, or choose **Service → Customize** from the main menu. Then select the check box beside each runlevel in which you want the service to run. Note that this option is not available for services controlled by **xinetd**.

12.2.2. Using the ntsysv Utility

The **ntsysv** utility is a command-line application with a simple text user interface to configure which services are to be started in selected runlevels. To start the utility, type **ntsysv** at a shell prompt as **root**.



Figure 12.2. The ntsysv utility

The utility displays the list of available services (the services from the `/etc/rc.d/init.d/` directory) along with their current status and a description obtainable by pressing **F1**. For a list of used symbols and an explanation of their meaning, see [Table 12.3, "Possible service states"](#).

Table 12.3. Possible service states

Symbol	Description
[*]	The service is enabled.
[]	The service is disabled.

12.2.2.1. Enabling and Disabling a Service

To enable a service, navigate through the list using the **Up** and **Down** arrows keys, and select it with the **Spacebar**. An asterisk (*) appears in the brackets.

To disable a service, navigate through the list using the **Up** and **Down** arrows keys, and toggle its status with the **Spacebar**. An asterisk (*) in the brackets disappears.

Once you are done, use the **Tab** key to navigate to the **Ok** button, and confirm the changes by pressing **Enter**. Keep in mind that `ntsysv` does not actually start or stop the service. If you need to start or stop the service immediately, use the `service` command as described in [Section 12.3.2, "Starting a Service"](#).

12.2.2.2. Selecting Runlevels

By default, the `ntsysv` utility only affects the current runlevel. To enable or disable services for other runlevels, as `root`, run the command with the additional `--level` option followed by numbers from 0 to 6 representing each runlevel you want to configure:

```
ntsysv --level runlevels
```

For example, to configure runlevels 3 and 5, type:

```
~]# ntsysv --level 35
```

12.2.3. Using the **chkconfig** Utility

The **chkconfig** utility is a command-line tool that allows you to specify in which runlevel to start a selected service, as well as to list all available services along with their current setting. Note that with the exception of listing, you must have superuser privileges to use this command.

12.2.3.1. Listing the Services

To display a list of system services (services from the **/etc/rc.d/init.d/** directory, as well as the services controlled by **xinetd**), either type **chkconfig --list**, or use **chkconfig** with no additional arguments. You will be presented with an output similar to the following:

```
~]# chkconfig --list
NetworkManager 0:off 1:off 2:on 3:on 4:on 5:on 6:off
abrt 0:off 1:off 2:off 3:on 4:off 5:on 6:off
acpid 0:off 1:off 2:on 3:on 4:on 5:on 6:off
anamon 0:off 1:off 2:off 3:off 4:off 5:off 6:off
atd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
auditd 0:off 1:off 2:on 3:on 4:on 5:on 6:off
avahi-daemon 0:off 1:off 2:off 3:on 4:on 5:on 6:off
... several lines omitted ...
wpa_supplicant 0:off 1:off 2:off 3:off 4:off 5:off 6:off

xinetd based services:
    chargen-dgram: off
    chargen-stream: off
    cvs: off
    daytime-dgram: off
    daytime-stream: off
    discard-dgram: off
    ... several lines omitted ...
    time-stream: off
```

Each line consists of the name of the service followed by its status (*on* or *off*) for each of the seven numbered runlevels. For example, in the listing above, **NetworkManager** is enabled in runlevel 2, 3, 4, and 5, while **abrt** runs in runlevel 3 and 5. The **xinetd** based services are listed at the end, being either *on*, or *off*.

To display the current settings for a selected service only, use **chkconfig --list** followed by the name of the service:

```
chkconfig --list service_name
```

For example, to display the current settings for the **sshd** service, type:

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

You can also use this command to display the status of a service that is managed by **xinetd**. In that case, the output will only contain the information whether the service is enabled or disabled:

```
~]# chkconfig --list rsync
rsync      off
```

12.2.3.2. Enabling a Service

To enable a service in runlevels 2, 3, 4, and 5, type the following at a shell prompt as **root**:

```
chkconfig service_name on
```

For example, to enable the **httpd** service in these four runlevels, type:

```
~]# chkconfig httpd on
```

To enable a service in certain runlevels only, add the **--level** option followed by numbers from 0 to 6 representing each runlevel in which you want the service to run:

```
chkconfig service_name on --level runlevels
```

For instance, to enable the **abrt** service in runlevels 3 and 5, type:

```
~]# chkconfig abrt on --level 35
```

The service will be started the next time you enter one of these runlevels. If you need to start the service immediately, use the **service** command as described in [Section 12.3.2, "Starting a Service"](#).

Do not use the **--level** option when working with a service that is managed by **xinetd**, as it is not supported. For example, to enable the **rsync** service, type:

```
~]# chkconfig rsync on
```

If the **xinetd** daemon is running, the service is immediately enabled without having to manually restart the daemon.

12.2.3.3. Disabling a Service

To disable a service in runlevels 2, 3, 4, and 5, type the following at a shell prompt as **root**:

```
chkconfig service_name off
```

For instance, to disable the **httpd** service in these four runlevels, type:

```
~]# chkconfig httpd off
```

To disable a service in certain runlevels only, add the **--level** option followed by numbers from 0 to 6 representing each runlevel in which you do *not* want the service to run:

```
chkconfig service_name off --level runlevels
```

For instance, to disable the **abrt** in runlevels 2 and 4, type:

```
~]# chkconfig abrtd off --level 24
```

The service will be stopped the next time you enter one of these runlevels. If you need to stop the service immediately, use the **service** command as described in [Section 12.3.3, “Stopping a Service”](#).

Do *not* use the **--level** option when working with a service that is managed by **xinetd**, as it is not supported. For example, to disable the **rsync** service, type:

```
~]# chkconfig rsync off
```

If the **xinetd** daemon is running, the service is immediately disabled without having to manually restart the daemon.

12.3. RUNNING SERVICES

The **service** utility allows you to start, stop, or restart the services from the **/etc/init.d/** directory.

12.3.1. Determining the Service Status

To determine the current status of a service, type the following at a shell prompt:

```
service service_name status
```

For example, to determine the status of the **httpd** service, type:

```
~]# service httpd status  
httpd (pid 7474) is running...
```

To display the status of all available services at once, run the **service** command with the **--status-all** option:

```
~]# service --status-all  
abrtd (pid 1492) is running...  
acpid (pid 1305) is running...  
atd (pid 1540) is running...  
audited (pid 1103) is running...  
automount (pid 1315) is running...  
Avahi daemon is running  
cpuspeed is stopped  
... several lines omitted ...  
wpa_supplicant (pid 1227) is running...
```

Note that you can also use the **Service Configuration** utility as described in [Section 12.2.1, “Using the Service Configuration Utility”](#).

12.3.2. Starting a Service

To start a service, type the following at a shell prompt as **root**:

```
service service_name start
```

For example, to start the **httpd** service, type:

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

12.3.3. Stopping a Service

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

```
service service_name stop
```

For example, to stop the **httpd** service, type:

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

12.3.4. Restarting a Service

To restart the service, type the following at a shell prompt as **root**:

```
service service_name restart
```

For example, to restart the **httpd** service, type:

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

12.4. ADDITIONAL RESOURCES

12.4.1. Installed Documentation

- **chkconfig(8)** – a manual page for the **chkconfig** utility.
- **ntsysv(8)** – a manual page for the **ntsysv** utility.
- **service(8)** – a manual page for the **service** utility.
- **system-config-services(8)** – a manual page for the **system-config-services** utility.

12.4.2. Related Books

Red Hat Enterprise Linux 6 Security Guide

A guide to securing Red Hat Enterprise Linux 6. It contains valuable information on how to set up the firewall, as well as the configuration of **SELinux**.

CHAPTER 13. CONFIGURING AUTHENTICATION

Authentication is the way that a user is identified and verified to a system. The authentication process requires presenting some sort of identity and credentials, like a user name and password. The credentials are then compared to information stored in some data store on the system. In Red Hat Enterprise Linux, the Authentication Configuration Tool helps configure what kind of data store to use for user credentials, such as LDAP.

For convenience and potentially part of single sign-on, Red Hat Enterprise Linux can use a central daemon to store user credentials for a number of different data stores. The System Security Services Daemon (SSSD) can interact with LDAP, Kerberos, and external applications to verify user credentials. The Authentication Configuration Tool can configure SSSD along with NIS, Winbind, and LDAP, so that authentication processing and caching can be combined.

13.1. CONFIGURING SYSTEM AUTHENTICATION

When a user logs into a Red Hat Enterprise Linux system, that user presents some sort of *credential* to establish the user identity. The system then checks those credentials against the configured authentication service. If the credentials match and the user account is active, then the user is *authenticated*. (Once a user is authenticated, then the information is passed to the access control service to determine what the user is permitted to do. Those are the resources the user is *authorized* to access.)

The information to verify the user can be located on the local system or the local system can reference a user database on a remote system, such as LDAP or Kerberos.

The system must have a configured list of valid account databases for it to check for user authentication. On Red Hat Enterprise Linux, the Authentication Configuration Tool has both GUI and command-line options to configure any user data stores.

A local system can use a variety of different data stores for user information, including Lightweight Directory Access Protocol (LDAP), Network Information Service (NIS), and Winbind. Additionally, both LDAP and NIS data stores can use Kerberos to authenticate users.

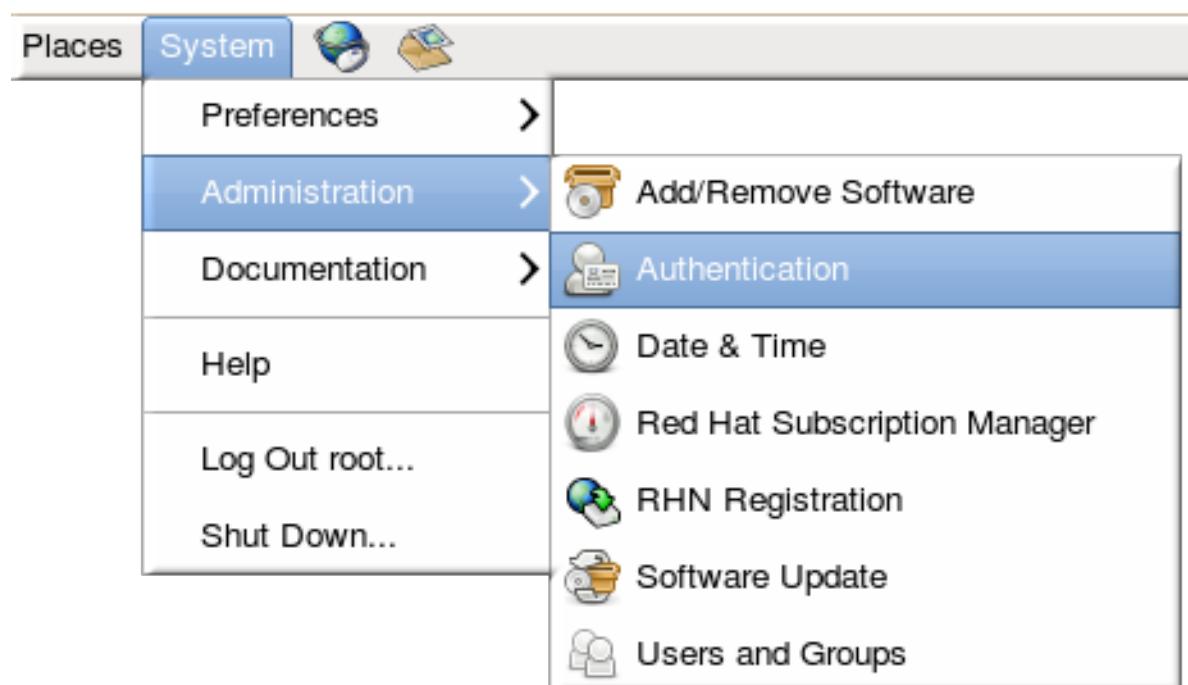


IMPORTANT

If a medium or high security level is set during installation or with the Security Level Configuration Tool, then the firewall prevents NIS authentication. For more information about firewalls, see the "Firewalls" section of the *Security Guide*.

13.1.1. Launching the Authentication Configuration Tool UI

1. Log into the system as root.
2. Open the **System**.
3. Select the **Administration** menu.
4. Select the **Authentication** item.



Alternatively, run the **system-config-authentication** command.



IMPORTANT

Any changes take effect immediately when the Authentication Configuration Tool UI is closed.

There are two configuration tabs in the **Authentication** dialog box:

- **Identity & Authentication**, which configures the resource used as the identity store (the data repository where the user IDs and corresponding credentials are stored).
- **Advanced Options**, which allows authentication methods other than passwords or certificates, like smart cards and fingerprint.

13.1.2. Selecting the Identity Store for Authentication

The **Identity & Authentication** tab sets how users should be authenticated. The default is to use local system authentication, meaning the users and their passwords are checked against local system accounts. A Red Hat Enterprise Linux machine can also use external resources which contain the users and credentials, including LDAP, NIS, and Winbind.

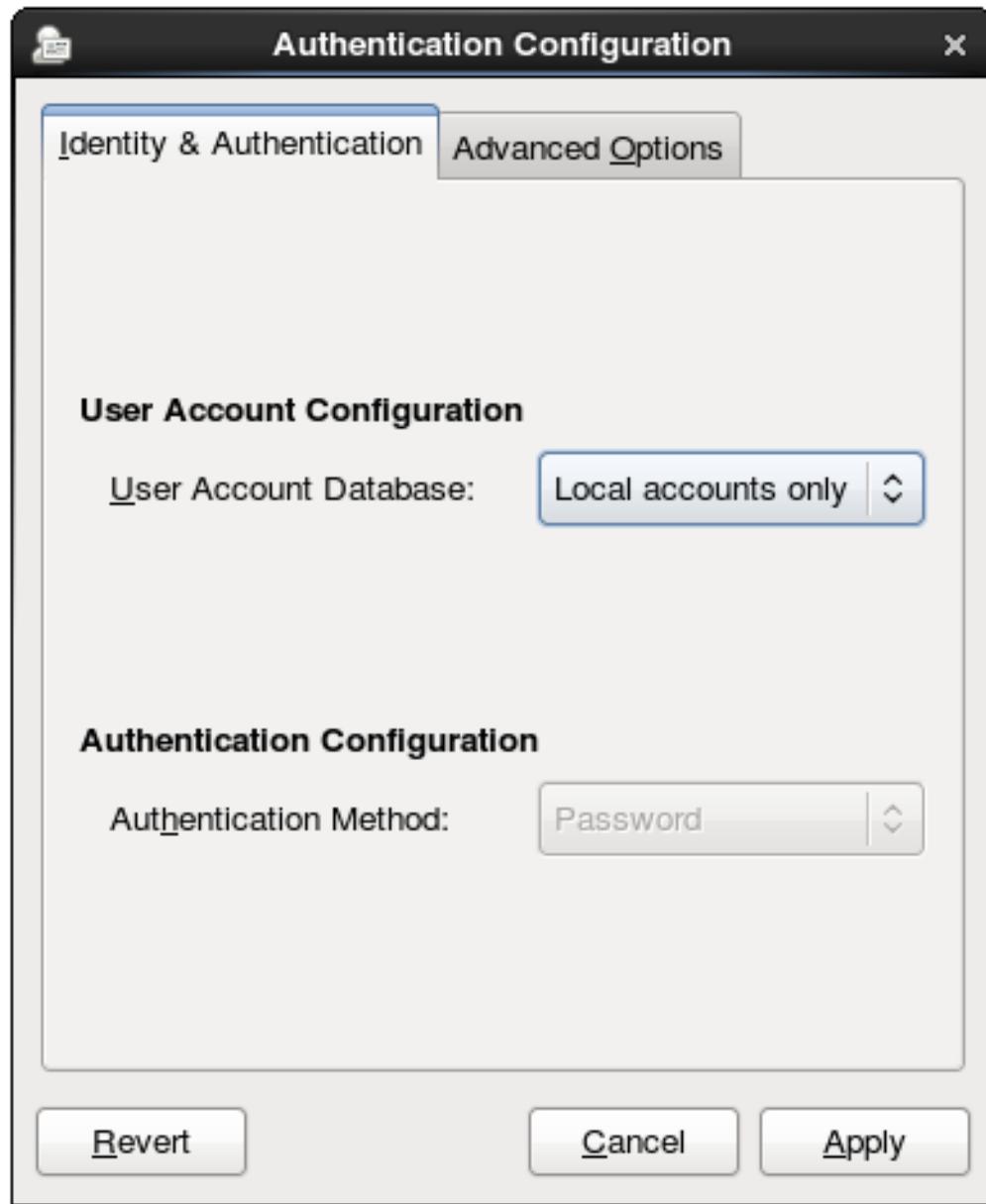
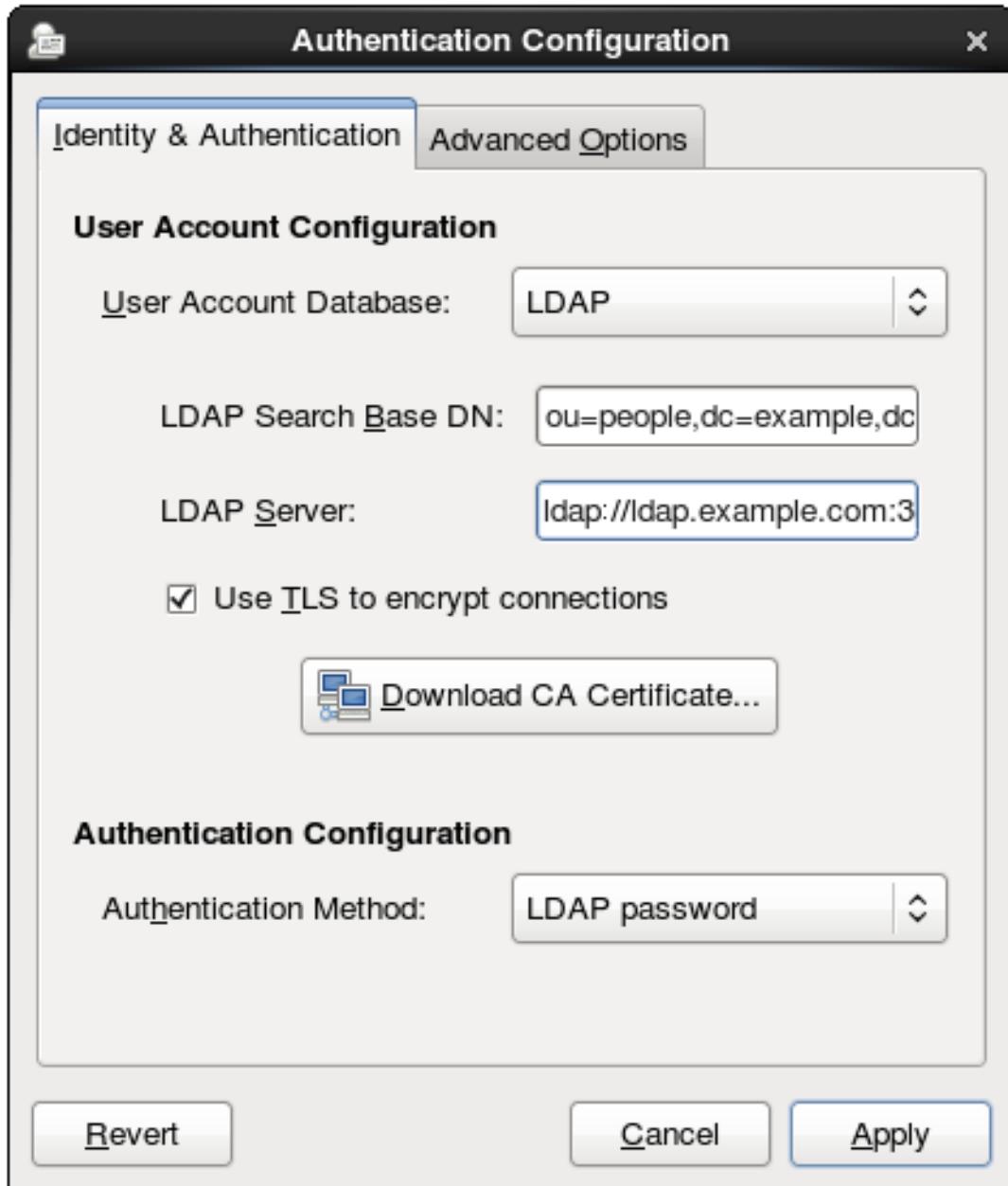


Figure 13.1. Local Authentication

13.1.2.1. Configuring LDAP Authentication

Either the `openldap-clients` package or the `sssd` package is used to configure an LDAP server for the user database. Both packages are installed by default.

1. Open the Authentication Configuration Tool, as in [Section 13.1.1, “Launching the Authentication Configuration Tool UI”](#).
2. Select **LDAP** in the **User Account Database** drop-down menu.



3. Set the information that is required to connect to the LDAP server.

- **LDAP Search Base DN** gives the root suffix or *distinguished name* (DN) for the user directory. All of the user entries used for identity/authentication will exist below this parent entry. For example, *ou=people,dc=example,dc=com*.

This field is optional. If it is not specified, then the System Security Services Daemon (SSSD) attempts to detect the search base using the ***namingContexts*** and ***defaultNamingContext*** attributes in the LDAP server's configuration entry.

- **LDAP Server** gives the URL of the LDAP server. This usually requires both the host name and port number of the LDAP server, such as *ldap://ldap.example.com:389*.

Entering the secure protocol in the URL, **ldaps://**, enables the **Download CA Certificate** button.

- **Use TLS to encrypt connections** sets whether to use Start TLS to encrypt the connections to the LDAP server. This enables a secure connection over a standard port.

Selecting TLS enables the **Download CA Certificate** button, which retrieves the issuing CA certificate for the LDAP server from whatever certificate authority issued it. The CA certificate must be in the privacy enhanced mail (PEM) format.



IMPORTANT

*Do not select **Use TLS to encrypt connections** if the server URL uses a secure protocol (**Idaps**). This option uses Start TLS, which initiates a secure connection over a standard port; if a secure port is specified, then a protocol like SSL must be used instead of Start TLS.*

4. Select the authentication method. LDAP allows simple password authentication or Kerberos authentication.

Using Kerberos is described in [Section 13.1.2.4, “Using Kerberos with LDAP or NIS Authentication”](#).

The **LDAP password** option uses PAM applications to use LDAP authentication. This option requires either a secure (**Idaps://**) URL or the TLS option to connect to the LDAP server.

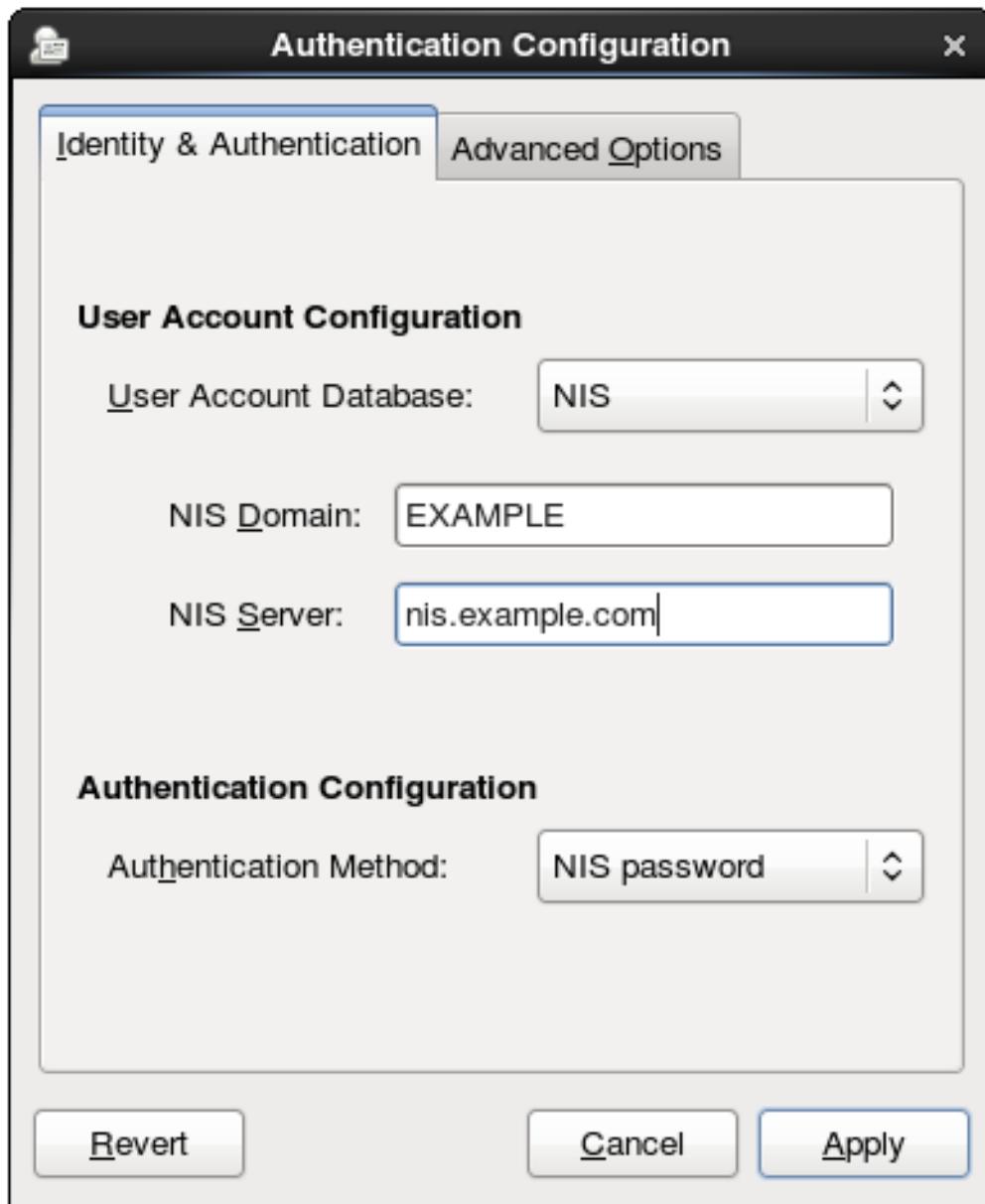
13.1.2.2. Configuring NIS Authentication

1. Install the **ypbind** package. This is required for NIS services, but is not installed by default.

```
~]# yum install ypbind
```

When the **ypbind** service is installed, the **portmap** and **ypbind** services are started and enabled to start at boot time.

2. Open the Authentication Configuration Tool, as in [Section 13.1.1, “Launching the Authentication Configuration Tool UI”](#).
3. Select **NIS** in the **User Account Database** drop-down menu.



- Set the information to connect to the NIS server, meaning the NIS domain name and the server host name. If the NIS server is not specified, the **authconfig** daemon scans for the NIS server.
- Select the authentication method. NIS allows simple password authentication or Kerberos authentication.

Using Kerberos is described in [Section 13.1.2.4, “Using Kerberos with LDAP or NIS Authentication”](#).

For more information about NIS, see the "Securing NIS" section of the *Security Guide*.

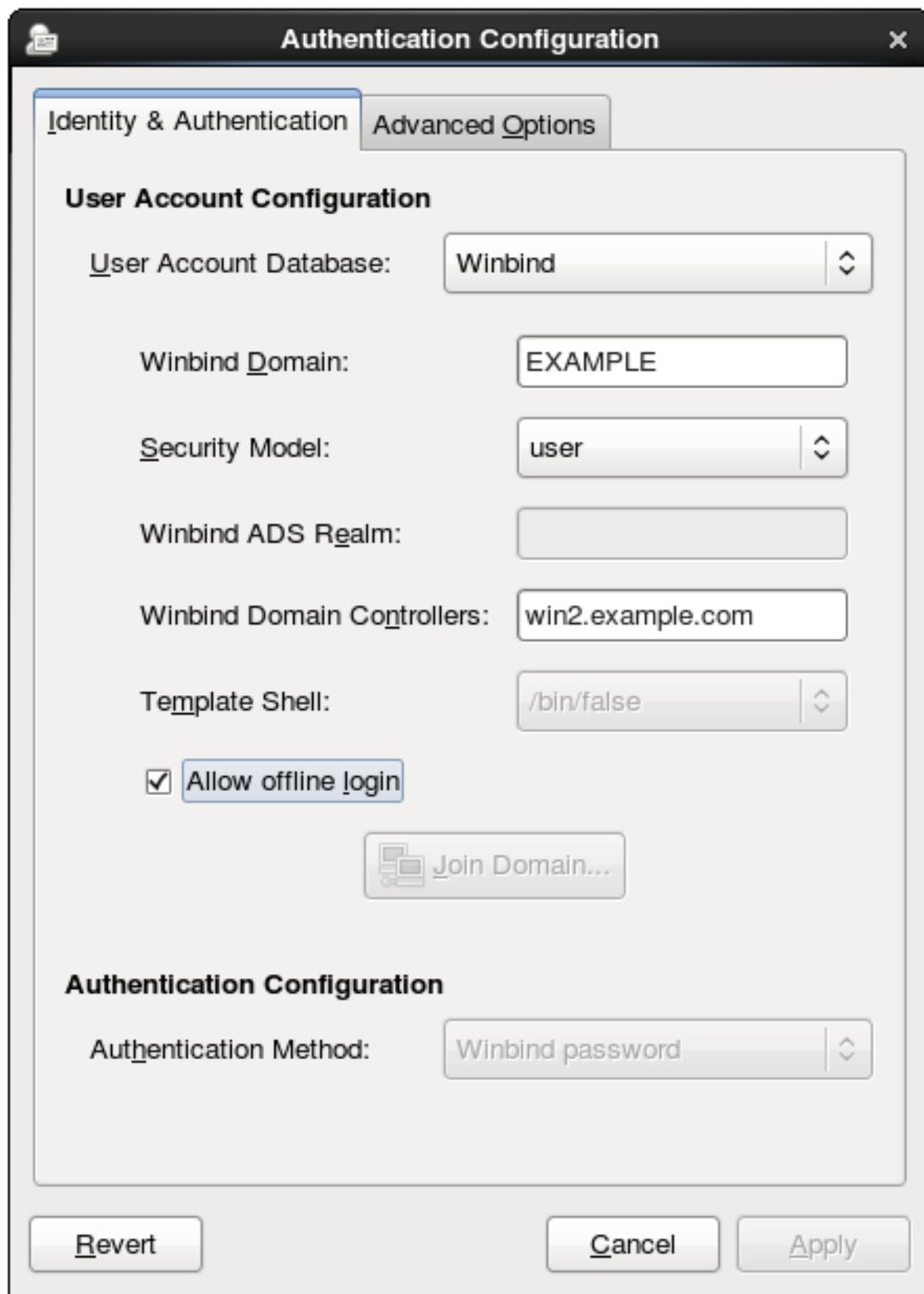
13.1.2.3. Configuring Winbind Authentication

- Install the samba-winbind package. This is required for Windows integration features in Samba services, but is not installed by default.

```
~]# yum install samba-winbind
```

- Open the Authentication Configuration Tool, as in [Section 13.1.1, “Launching the Authentication Configuration Tool UI”](#).

3. Select Winbind in the **User Account Database** drop-down menu.



4. Set the information that is required to connect to the Microsoft Active Directory domain controller.
- **Winbind Domain** gives the Windows domain to connect to.
This should be in the Windows 2000 format, such as **DOMAIN**.
 - **Security Model** sets the security model to use for Samba clients. **authconfig** supports four types of security models:

- **ads** configures Samba to act as a domain member in an Active Directory Server realm. To operate in this mode, the krb5-server package must be installed and Kerberos must be configured properly. Also, when joining to the Active Directory Server using the command line, the following command must be used:

net ads join

- **domain** has Samba validate the user name/password by authenticating it through a Windows primary or backup domain controller, much like a Windows server.
- **server** has a local Samba server validate the user name/password by authenticating it through another server, such as a Windows server. If the server authentication attempt fails, the system then attempts to authenticate using **user** mode.
- **user** requires a client to log in with a valid user name and password. This mode does support encrypted passwords.

The user name format must be *domain\user*, such as **EXAMPLE\jsmith**.



NOTE

When verifying that a given user exists in the Windows domain, always use Windows 2000-style formats and escape the backslash (\) character. For example:

```
~]# getent passwd domain\\user
DOMAIN\\user:*:16777216:16777216:Name
Surname:/home/DOMAIN/user:/bin/bash
```

This is the default option.

- **Winbind ADS Realm** gives the Active Directory realm that the Samba server will join. This is only used with the **ads** security model.
- **Winbind Domain Controllers** gives the domain controller to use. For more information about domain controllers, see [Section 21.1.6.3, “Domain Controller”](#).
- **Template Shell** sets which login shell to use for Windows user account settings.
- **Allow offline login** allows authentication information to be stored in a local cache. The cache is referenced when a user attempts to authenticate to system resources while the system is offline.

For more information about the **Winbind** service, see [Section 21.1.2, “Samba Daemons and Related Services”](#).

For additional information about configuring **Winbind** and troubleshooting tips, see the [Knowledgebase](#) on the [Red Hat Customer Portal](#).

Also, the [Red Hat Access Labs](#) page includes the **Winbind Mapper** utility that generates a part of the **smb.conf** file to help you connect a Red Hat Enterprise Linux to an Active Directory.

13.1.2.4. Using Kerberos with LDAP or NIS Authentication

Both LDAP and NIS authentication stores support Kerberos authentication methods. Using Kerberos has a couple of benefits:

- It uses a security layer for communication while still allowing connections over standard ports.
- It automatically uses credentials caching with SSSD, which allows offline logins.

Using Kerberos authentication requires the krb5-libs and krb5-workstation packages.

The **Kerberos password** option from the **Authentication Method** drop-down menu automatically opens the fields required to connect to the Kerberos realm.

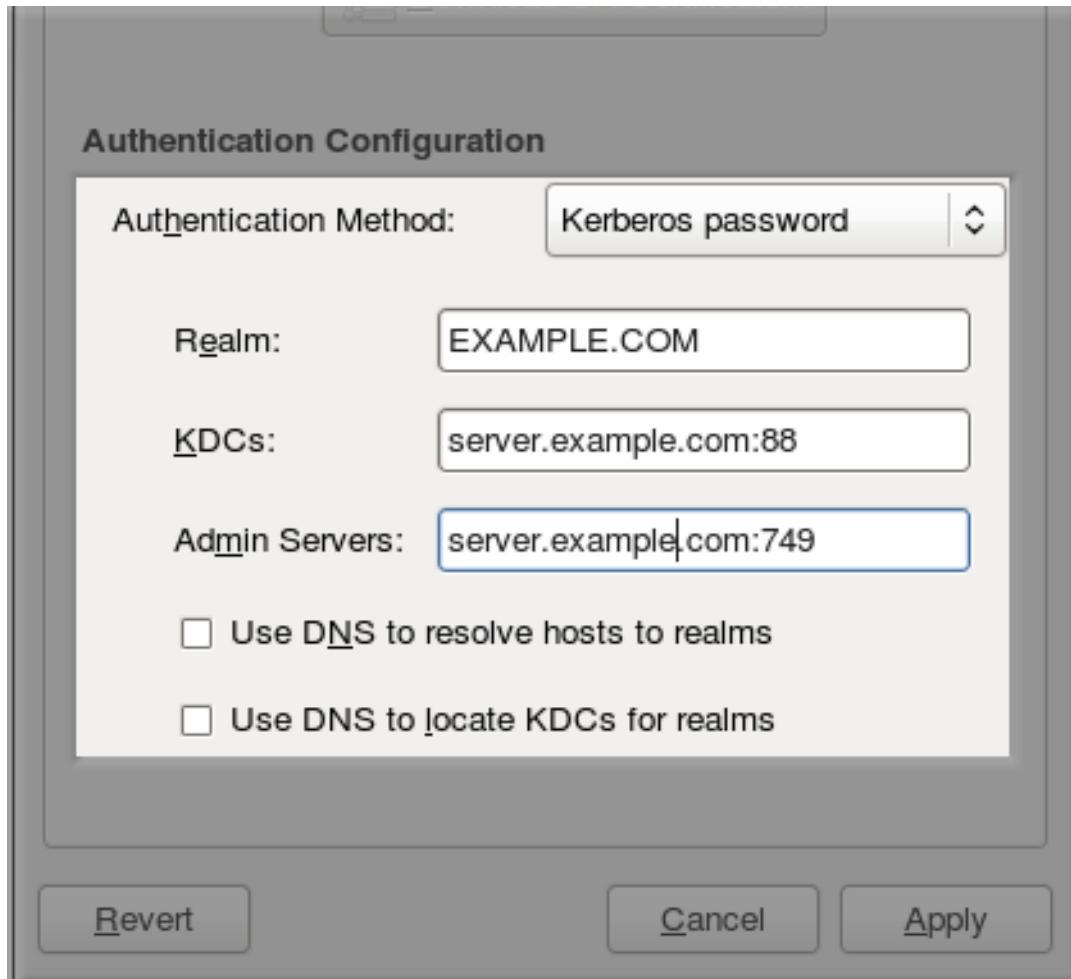


Figure 13.2. Kerberos Fields

- **Realm** gives the name for the realm for the Kerberos server. The realm is the network that uses Kerberos, composed of one or more *key distribution centers* (KDC) and a potentially large number of clients.
- **KDCs** gives a comma-separated list of servers that issue Kerberos tickets.
- **Admin Servers** gives a list of administration servers running the **kadmind** process in the realm.
- Optionally, use DNS to resolve server host name and to find additional KDCs within the realm.

For more information about Kerberos, see section "Using Kerberos" of the Red Hat Enterprise Linux 6 *Managing Single Sign-On and Smart Cards* guide.

13.1.3. Configuring Alternative Authentication Features

The Authentication Configuration Tool also configures settings related to authentication behavior, apart from the identity store. This includes entirely different authentication methods (fingerprint scans and smart cards) or local authentication rules. These alternative authentication options are configured in the **Advanced Options** tab.

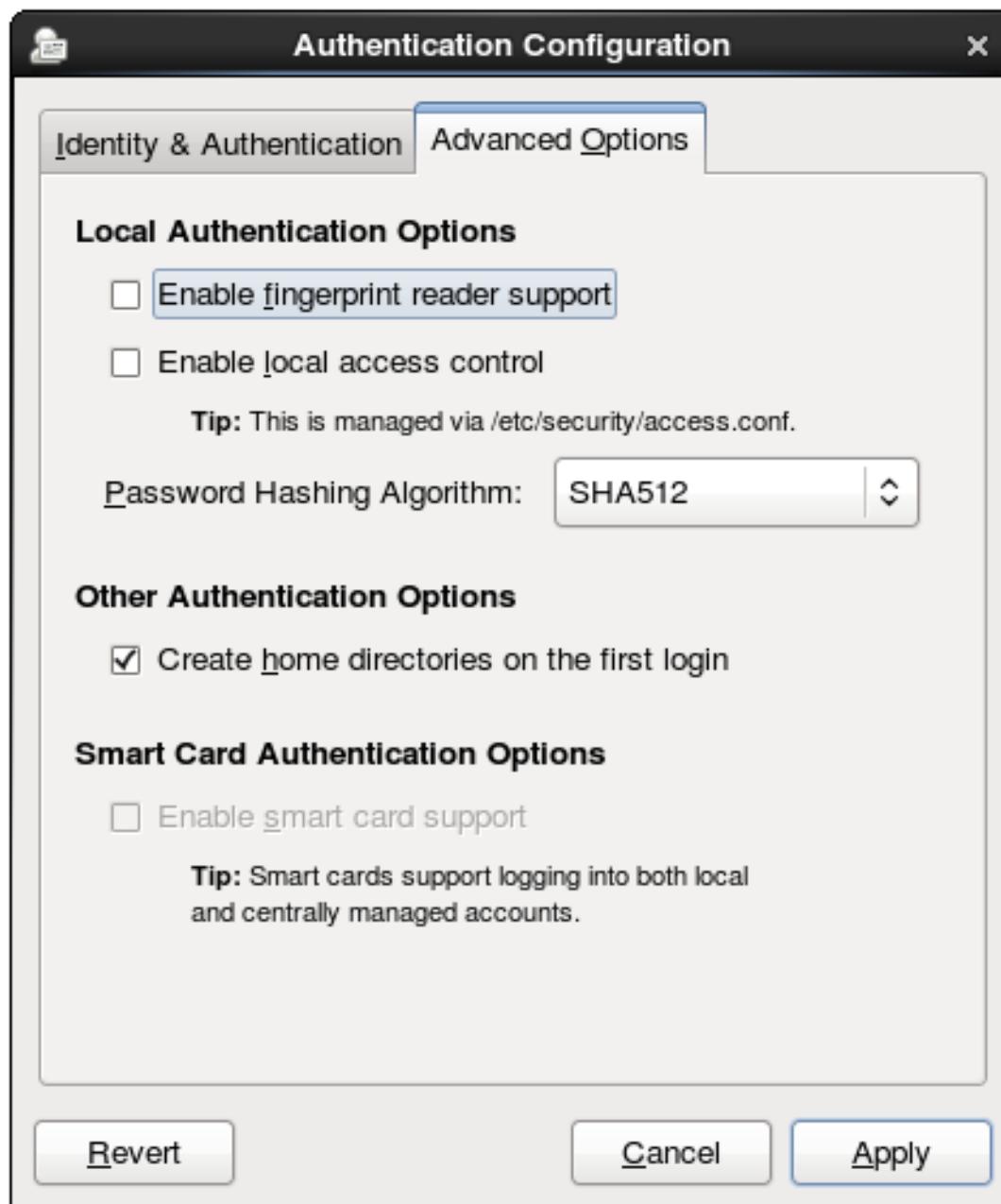


Figure 13.3. Advanced Options

13.1.3.1. Using Fingerprint Authentication

When there is appropriate hardware available, the **Enable fingerprint reader support** option allows fingerprint scans to be used to authenticate local users in addition to other credentials.

13.1.3.2. Setting Local Authentication Parameters

There are two options in the **Local Authentication Options** area which define authentication behavior on the local system:

- **Enable local access control** instructs the `/etc/security/access.conf` file to check for local user authorization rules.

- **Password Hashing Algorithm** sets the hashing algorithm to use to encrypt locally-stored passwords.

13.1.3.3. Enabling Smart Card Authentication

When there are appropriate smart card readers available, a system can accept smart cards (or *tokens*) instead of other user credentials to authenticate.

Once the **Enable smart card support** option is selected, then the behaviors of smart card authentication can be defined:

- **Card Removal Action** tells the system how to respond when the card is removed from the card reader during an active session. A system can either ignore the removal and allow the user to access resources as normal, or a system can immediately lock until the smart card is supplied.
- **Require smart card login** sets whether a smart card is required for logins or allowed for logins. When this option is selected, all other methods of authentication are immediately blocked.



WARNING

Do not select this option until you have successfully authenticated to the system using a smart card.

Using smart cards requires the pam_pkcs11 package.

13.1.3.4. Creating User Home Directories

There is an option (**Create home directories on the first login**) to create a home directory automatically the first time that a user logs in.

This option is beneficial with accounts that are managed centrally, such as with LDAP. However, this option should not be selected if a system like automount is used to manage user home directories.

13.1.4. Configuring Authentication from the Command Line

The **authconfig** command-line tool updates all of the configuration files and services required for system authentication, according to the settings passed to the script. Along with allowing all of the identity and authentication configuration options that can be set through the UI, the **authconfig** tool can also be used to create backup and kickstart files.

For a complete list of **authconfig** options, check the help output and the man page.

13.1.4.1. Tips for Using authconfig

There are some things to remember when running **authconfig**:

- With every command, use either the **--update** or **--test** option. One of those options is required for the command to run successfully. Using **--update** writes the configuration changes. **--test** prints the changes to stdout but does not apply the changes to the configuration.

- Each enable option has a corresponding disable option.

13.1.4.2. Configuring LDAP User Stores

To use an LDAP identity store, use the **--enableldap**. To use LDAP as the authentication source, use **--enableldapauth** and then the requisite connection information, like the LDAP server name, base DN for the user suffix, and (optionally) whether to use TLS. The **authconfig** command also has options to enable or disable RFC 2307bis schema for user entries, which is not possible through the Authentication Configuration UI.

Be sure to use the full LDAP URL, including the protocol (**ldap** or **ldaps**) and the port number. Do *not* use a secure LDAP URL (**ldaps**) with the **--enableldaptls** option.

```
authconfig --enableldap --enableldapauth --
ldapserver=ldap://ldap.example.com:389,ldap://ldap2.example.com:389 --
ldapbasedn="ou=people,dc=example,dc=com" --enableldaptls --
ldaploadcacert=https://ca.server.example.com/caCert.crt --update
```

Instead of using **--ldapauth** for LDAP password authentication, it is possible to use Kerberos with the LDAP user store. These options are described in [Section 13.1.4.5, “Configuring Kerberos Authentication”](#).

13.1.4.3. Configuring NIS User Stores

To use a NIS identity store, use the **--enablenis**. This automatically uses NIS authentication, unless the Kerberos parameters are explicitly set, so it uses Kerberos authentication ([Section 13.1.4.5, “Configuring Kerberos Authentication”](#)). The only parameters are to identify the NIS server and NIS domain; if these are not used, then the **authconfig** service scans the network for NIS servers.

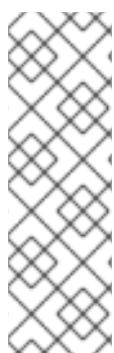
```
authconfig --enablenis --nisdomain=EXAMPLE --nisserver=nis.example.com --update
```

13.1.4.4. Configuring Winbind User Stores

Windows domains have several different security models, and the security model used in the domain determines the authentication configuration for the local system.

For user and server security models, the Winbind configuration requires only the domain (or workgroup) name and the domain controller host names.

```
authconfig --enablewinbind --enablewinbindauth --smbsecurity=user|server --enablewinbindoffline --
smbservers=ad.example.com --smbworkgroup=EXAMPLE --update
```



NOTE

The user name format must be *domain\user*, such as **EXAMPLEjsmith**.

When verifying that a given user exists in the Windows domain, always use Windows 2000-style formats and escape the backslash (\) character. For example:

```
~]# getent passwd domain\user DOMAIN\user:*:16777216:16777216:Name
Surname:/home/DOMAIN/user:/bin/bash
```

For ads and domain security models, the Winbind configuration allows additional configuration for the template shell and realm (ads only). For example:

```
authconfig --enablewinbind --enablewinbindauth --smbsecurity ads --enablewinbindoffline --smbservers=ad.example.com --smbworkgroup=EXAMPLE --smbrealm EXAMPLE.COM --winbindtemplateshell=/bin/sh --update
```

There are a lot of other options for configuring Windows-based authentication and the information for Windows user accounts, such as name formats, whether to require the domain name with the user name, and UID ranges. These options are listed in the **authconfig** help.

13.1.4.5. Configuring Kerberos Authentication

Both LDAP and NIS allow Kerberos authentication to be used in place of their native authentication mechanisms. At a minimum, using Kerberos authentication requires specifying the realm, the KDC, and the administrative server. There are also options to use DNS to resolve client names and to find additional admin servers.

```
authconfig NIS or LDAP options --enablekrb5 --krb5realm EXAMPLE --krb5kdc kdc.example.com:88,server.example.com:88 --krb5adminserver server.example.com:749 --enablekrb5kcdns --enablekrb5realmdns --update
```

13.1.4.6. Configuring Local Authentication Settings

The Authentication Configuration Tool can also control some user settings that relate to security, such as creating home directories, setting password hash algorithms, and authorization. These settings are done independently of identity/user store settings.

For example, to create user home directories:

```
authconfig --enablemkhomedir --update
```

To set or change the hash algorithm used to encrypt user passwords:

```
authconfig --passalgo=sha512 --update
```

13.1.4.7. Configuring Fingerprint Authentication

There is one option to enable support for fingerprint readers. This option can be used alone or in conjunction with other **authconfig** settings, like LDAP user stores.

```
~]# authconfig --enablefingerprint --update
```

13.1.4.8. Configuring Smart Card Authentication

All that is required to use smart cards with a system is to set the **--enablesmartcard** option:

```
~]# authconfig --enablesmartcard --update
```

There are other configuration options for smart cards, such as changing the default smart card module, setting the behavior of the system when the smart card is removed, and requiring smart cards for login.

For example, this command instructs the system to lock out a user immediately if the smart card is removed (a setting of 1 ignores it if the smart card is removed):

```
~]# authconfig --enablesmartcard --smartcardaction=0 --update
```

Once smart card authentication has been successfully configured and tested, then the system can be configured to require smart card authentication for users rather than simple password-based authentication.

```
~]# authconfig --enablerequiresmartcard --update
```



WARNING

Do not use the **--enablerequiresmartcard** option until you have successfully authenticated to the system using a smart card. Otherwise, users may be unable to log into the system.

13.1.4.9. Managing Kickstart and Configuration Files

The **--update** option updates all of the configuration files with the configuration changes. There are a couple of alternative options with slightly different behavior:

- **--kickstart** writes the updated configuration to a kickstart file.
- **--test** prints the full configuration, with changes, to stdout but does not edit any configuration files.

Additionally, **authconfig** can be used to back up and restore previous configurations. All archives are saved to a unique subdirectory in the **/var/lib/authconfig/** directory. For example, the **--savebackup** option gives the backup directory as **2011-07-01**:

```
~]# authconfig --savebackup=2011-07-01
```

This backs up all of the authentication configuration files beneath the **/var/lib/authconfig/backup-2011-07-01** directory.

Any of the saved backups can be used to restore the configuration using the **--restorebackup** option, giving the name of the manually-saved configuration:

```
~]# authconfig --restorebackup=2011-07-01
```

Additionally, **authconfig** automatically makes a backup of the configuration before it applies any changes (with the **--update** option). The configuration can be restored from the most recent automatic backup, without having to specify the exact backup, using the **--restorelastbackup** option.

13.1.5. Using Custom Home Directories

If LDAP users have home directories that are not in **/home** and the system is configured to create home directories the first time users log in, then these directories are created with the wrong permissions.

1. Apply the correct SELinux context and permissions from the **/home** directory to the home directory that is created on the local system. For example:

```
~]# semanage fcontext -a -e /home /home/locale
```

2. Install the oddjob-mkhomedir package on the system.

This package provides the **pam_oddjob_mkhomedir.so** library, which the Authentication Configuration Tool uses to create home directories. The **pam_oddjob_mkhomedir.so** library, unlike the default **pam_mkhomedir.so** library, can create SELinux labels.

The Authentication Configuration Tool automatically uses the **pam_oddjob_mkhomedir.so** library if it is available. Otherwise, it will default to using **pam_mkhomedir.so**.

3. Make sure the **oddjobd** service is running.
4. Re-run the Authentication Configuration Tool and enable home directories, as in [Section 13.1.3, "Configuring Alternative Authentication Features"](#).

If home directories were created before the home directory configuration was changed, then correct the permissions and SELinux contexts. For example:

```
~]# semanage fcontext -a -e /home /home/locale  
# restorecon -R -v /home/locale
```

13.2. USING AND CACHING CREDENTIALS WITH SSSD

The System Security Services Daemon (SSSD) provides access to different identity and authentication providers.

13.2.1. About SSSD

Most system authentication is configured locally, which means that services must check with a local user store to determine users and credentials. What SSSD does is allow a local service to check with a local cache in SSSD, but that cache may be taken from any variety of *remote* identity providers – an LDAP directory, an Identity Management domain, Active Directory, possibly even a Kerberos realm.

SSSD also caches those users and credentials, so if the local system or the identity provider go offline, the user credentials are still available to services to verify.

SSSD is an intermediary between local clients and any configured data store. This relationship brings a number of benefits for administrators:

- *Reducing the load on identification/authentication servers.* Rather than having every client service attempt to contact the identification server directly, all of the local clients can contact SSSD which can connect to the identification server or check its cache.
- *Permitting offline authentication.* SSSD can optionally keep a cache of user identities and credentials that it retrieves from remote services. This allows users to authenticate to resources successfully, even if the remote identification server is offline or the local machine is offline.
- *Using a single user account.* Remote users frequently have two (or even more) user accounts, such as one for their local system and one for the organizational system. This is necessary to connect to a virtual private network (VPN). Because SSSD supports caching and offline

authentication, remote users can connect to network resources by authenticating to their local machine and then SSSD maintains their network credentials.

Additional Resources

While this chapter covers the basics of configuring services and domains in SSSD, this is not a comprehensive resource. Many other configuration options are available for each functional area in SSSD; check out the man page for the specific functional area to get a complete list of options.

Some of the common man pages are listed in [Table 13.1, “A Sampling of SSSD Man Pages”](#). There is also a complete list of SSSD man pages in the “See Also” section of the **sssd(8)** man page.

Table 13.1. A Sampling of SSSD Man Pages

Functional Area	Man Page
General Configuration	<code>sssd.conf(8)</code>
sudo Services	<code>sssd-sudo</code>
LDAP Domains	<code>sssd-ldap</code>
Active Directory Domains	<code>sssd-ad</code> <code>sssd-ldap</code>
Identity Management (IdM or IPA) Domains	<code>sssd-ipa</code> <code>sssd-ldap</code>
Kerberos Authentication for Domains	<code>sssd-krb5</code>
OpenSSH Keys	<code>sss_ssh_authorizedkeys</code> <code>sss_ssh_knownhostsproxy</code>
Cache Maintenance	<code>sss_cache (cleanup)</code> <code>sss_useradd, sss_usermod, sss_userdel, sss_seed</code> (user cache entry management)

13.2.2. Setting up the `sssd.conf` File

SSSD services and domains are configured in a `.conf` file. By default, this is `/etc/sssd/sssd.conf` – although that file must be created and configured manually, since SSSD is not configured after installation.

13.2.2.1. Creating the `sssd.conf` File

There are three parts of the SSSD configuration file:

- **[sssd]**, for general SSSD process and operational configuration; this basically lists the configured services, domains, and configuration parameters for each
- **[service_name]**, for configuration options for each supported system service, as described in [Section 13.2.4, “SSSD and System Services”](#)
- **[domain_type/DOMAIN_NAME]**, for configuration options for each configured identity provider



IMPORTANT

While services are optional, at least one identity provider domain must be configured before the SSSD service can be started.

Example 13.1. Simple `sssd.conf` File

```
[sssd]
domains = LOCAL
services = nss
config_file_version = 2

[nss]
filter_groups = root
filter_users = root

[domain/LOCAL]
id_provider = local
auth_provider = local
access_provider = permit
```

The **[sssd]** section has three important parameters:

- **domains** lists all of the domains, configured in the **sssd.conf**, which SSSD uses as identity providers. If a domain is not listed in the **domains** key, it is not used by SSSD, even if it has a configuration section.
- **services** lists all of the system services, configured in the **sssd.conf**, which use SSSD; when SSSD starts, the corresponding SSSD service is started for each configured system service. If a service is not listed in the **services** key, it is not used by SSSD, even if it has a configuration section.
- **config_file_version** sets the version of the configuration file to set file format expectations. This is version 2, for all recent SSSD versions.



NOTE

Even if a service or domain is configured in the **sssd.conf** file, SSSD does not interact with that service or domain unless it is listed in the **services** or **domains** parameters, respectively, in the **[sssd]** section.

Other configuration parameters are listed in the **sssd.conf** man page.

Each service and domain parameter is described in its respective configuration section in this chapter and in their man pages.

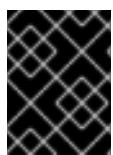
13.2.2.2. Using a Custom Configuration File

By default, the **sssd** process assumes that the configuration file is **/etc/sssd/sssd.conf**.

An alternative file can be passed to SSSD by using the **-c** option with the **sssd** command:

```
~]# sssd -c /etc/sssd/customfile.conf --daemon
```

13.2.3. Starting and Stopping SSSD



IMPORTANT

Configure at least one domain before starting SSSD for the first time. See [Section 13.2.10, “SSSD and Identity Providers \(Domains\)”](#).

Either the **service** command or the **/etc/init.d/sssd** script can start SSSD. For example:

```
~]# service sssd start
```

By default, SSSD is not configured to start automatically. There are two ways to change this behavior:

- Enabling SSSD through the **authconfig** command:

```
~]# authconfig --enablesssd --enablesssdauth --update
```

- Adding the SSSD process to the start list using the **chkconfig** command:

```
~]# chkconfig sssd on
```

13.2.4. SSSD and System Services

SSSD and its associated services are configured in the **sssd.conf** file. The **[sssd]** section also lists the services that are active and should be started when **sssd** starts within the **services** directive.

SSSD can provide credentials caches for several system services:

- A Name Service Switch (NSS) provider service that answers name service requests from the **sssd_nss** module. This is configured in the **[nss]** section of the SSSD configuration.

This is described in [Section 13.2.5, “Configuring Services: NSS”](#).

- A PAM provider service that manages a PAM conversation through the **sssd_pam** module. This is configured in the **[pam]** section of the configuration.

This is described in [Section 13.2.6, “Configuring Services: PAM”](#).

- An SSH provider service that defines how SSSD manages the **known_hosts** file and other key-related configuration. Using SSSD with OpenSSH is described in [Section 13.2.9, “Configuring Services: OpenSSH and Cached Keys”](#).
- An **autofs** provider service that connects to an LDAP server to retrieve configured mount locations. This is configured as part of an LDAP identity provider in a **[domain/NAME]** section in the configuration file.

This is described in [Section 13.2.7, “Configuring Services: autofs”](#).

- A **sudo** provider service that connects to an LDAP server to retrieve configured **sudo** policies. This is configured as part of an LDAP identity provider in a **[domain/NAME]** section in the configuration file.

This is described in [Section 13.2.8, “Configuring Services: sudo”](#).

- A PAC responder service that defines how SSSD works with Kerberos to manage Active Directory users and groups. This is specifically part of managing Active Directory identity providers with domains, as described in [Section 13.2.13, “Creating Domains: Active Directory”](#).

13.2.5. Configuring Services: NSS

SSSD provides an NSS module, **sssd_nss**, which instructs the system to use SSSD to retrieve user information. The NSS configuration must include a reference to the SSSD module, and then the SSSD configuration sets how SSSD interacts with NSS.

About NSS Service Maps and SSSD

The Name Service Switch (NSS) provides a central configuration for services to look up a number of configuration and name resolution services. NSS provides one method of mapping system identities and services with configuration sources.

SSSD works with NSS as a provider services for several types of NSS maps:

- Passwords (**passwd**)
- User groups (**shadow**)
- Groups (**groups**)
- Netgroups (**netgroups**)
- Services (**services**)

Procedure 13.1. Configuring NSS Services to Use SSSD

NSS can use multiple identity and configuration providers for any and all of its service maps. The default is to use system files for services; for SSSD to be included, the **nss_sss** module has to be included for the desired service type.

1. Use the Authentication Configuration tool to enable SSSD. This automatically configured the **nsswitch.conf** file to use SSSD as a provider.

```
~]# authconfig --enablesssd --update
```

This automatically configures the password, shadow, group, and netgroups services maps to use the SSSD module:

```

passwd: files sss
shadow: files sss
group: files sss

netgroup: files sss

```

- The services map is not enabled by default when SSSD is enabled with **authconfig**. To include that map, open the **nsswitch.conf** file and add the **sss** module to the **services** map:

```

~]# vim /etc/nsswitch.conf

...
services: file sss
...

```

Procedure 13.2. Configuring SSSD to Work with NSS

The options and configuration that SSSD uses to service NSS requests are configured in the SSSD configuration file, in the **[nss]** services section.

- Open the **sssd.conf** file.

```

~]# vim /etc/sssd/sssd.conf

```

- Make sure that NSS is listed as one of the services that works with SSSD.

```

[sssd]
config_file_version = 2
reconnection_retries = 3
sbus_timeout = 30
services = nss, pam

```

- In the **[nss]** section, change any of the NSS parameters. These are listed in [Table 13.2, “SSSD \[nss\] Configuration Parameters”](#).

```

[nss]
filter_groups = root
filter_users = root
reconnection_retries = 3
entry_cache_timeout = 300
entry_cache_nowait_percentage = 75

```

- Restart SSSD.

```

~]# service sssd restart

```

Table 13.2. SSSD [nss] Configuration Parameters

Parameter	Value Format	Description
entry_cache_nowait_percent age	integer	<p>Specifies how long sssd_nss should return cached entries before refreshing the cache. Setting this to zero (0) disables the entry cache refresh.</p> <p>This configures the entry cache to update entries in the background automatically if they are requested if the time before the next update is a certain percentage of the next interval. For example, if the interval is 300 seconds and the cache percentage is 75, then the entry cache will begin refreshing when a request comes in at 225 seconds – 75% of the interval.</p> <p>The allowed values for this option are 0 to 99, which sets the percentage based on the entry_cache_timeout value. The default value is 50%.</p>
entry_negative_timeout	integer	Specifies how long, in seconds, sssd_nss should cache negative cache hits. A negative cache hit is a query for an invalid database entries, including non-existent entries.
filter_users, filter_groups	string	Tells SSSD to exclude certain users from being fetched from the NSS database. This is particularly useful for system accounts such as root .
filter_users_in_groups	Boolean	Sets whether users listed in the filter_users list appear in group memberships when performing group lookups. If set to FALSE , group lookups return all users that are members of that group. If not specified, this value defaults to true , which filters the group member lists.
debug_level	integer, 0 – 9	Sets a debug logging level.

NSS Compatibility Mode

NSS compatibility (compat) mode provides the support for additional entries in the **/etc/passwd** file to ensure that users or members of netgroups have access to the system.

To enable NSS compatibility mode to work with SSSD, add the following entries to the **/etc/nsswitch.conf** file:

```
passwd: compat
passwd_compat: sss
```

Once NSS compatibility mode is enabled, the following **passwd** entries are supported:

- **+user-user**

Include (+) or exclude (-) a specified *user* from the Network Information System (NIS) map.

- **+@netgroup -@netgroup**

Include (+) or exclude (-) all users in the given *netgroup* from the NIS map.

- +

Exclude all users, except previously excluded ones from the NIS map.

For more information about NSS compatibility mode, see the **nsswitch.conf(5)** manual page.

13.2.6. Configuring Services: PAM



WARNING

A mistake in the PAM configuration file can lock users out of the system completely. Always back up the configuration files before performing any changes, and keep a session open so that any changes can be reverted.

SSSD provides a PAM module, **sssd_pam**, which instructs the system to use SSSD to retrieve user information. The PAM configuration must include a reference to the SSSD module, and then the SSSD configuration sets how SSSD interacts with PAM.

Procedure 13.3. Configuring PAM

1. Use **authconfig** to enable SSSD for system authentication.

```
# authconfig --update --enablerssd --enablerssdauth
```

This automatically updates the PAM configuration to reference all of the SSSD modules:

```
%#PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth    required    pam_env.so
auth    sufficient  pam_unix.so nullok try_first_pass
auth    requisite    pam_succeed_if.so uid >= 500 quiet
auth sufficient pam_sss.so use_first_pass
auth    required    pam_deny.so

account   required    pam_unix.so
account   sufficient  pam_localuser.so
account   sufficient  pam_succeed_if.so uid < 500 quiet
account [default=bad success=ok user_unknown=ignore] pam_sss.so
account   required    pam_permit.so

password  requisite    pam_cracklib.so try_first_pass retry=3
password  sufficient  pam_unix.so sha512 shadow nullok try_first_pass use_authtok
password sufficient pam_sss.so use_authtok
password  required    pam_deny.so

session   optional    pam_keyinit.so revoke
```

```

session required pam_limits.so
session [success=1 default=ignore] pam_succeed_if.so service in crond quiet use_uid
session sufficient pam_sss.so
session required pam_unix.so

```

These modules can be set to **include** statements, as necessary.

2. Open the **sssd.conf** file.

```
# vim /etc/sssd/sssd.conf
```

3. Make sure that PAM is listed as one of the services that works with SSSD.

```

[sssd]
config_file_version = 2
reconnection_retries = 3
sbus_timeout = 30
services = nss, pam

```

4. In the **[pam]** section, change any of the PAM parameters. These are listed in [Table 13.3, "SSSD \[pam\] Configuration Parameters"](#).

```

[pam]
reconnection_retries = 3
offline_credentials_expiration = 2
offline_failed_login_attempts = 3
offline_failed_login_delay = 5

```

5. Restart SSSD.

```
~]# service sssd restart
```

Table 13.3. SSSD [pam] Configuration Parameters

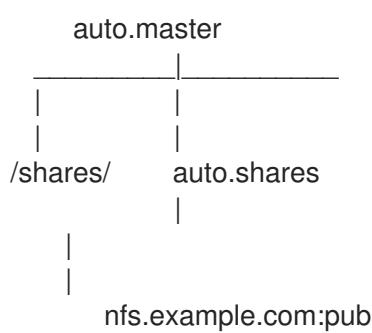
Parameter	Value Format	Description
offline_credentials_expiration	integer	Sets how long, in days, to allow cached logins if the authentication provider is offline. This value is measured from the last successful online login. If not specified, this defaults to zero (0), which is unlimited.
offline_failed_login_attempts	integer	Sets how many failed login attempts are allowed if the authentication provider is offline. If not specified, this defaults to zero (0), which is unlimited.
offline_failed_login_delay	integer	Sets how long to prevent login attempts if a user hits the failed login attempt limit. If set to zero (0), the user cannot authenticate while the provider is offline once he hits the failed attempt limit. Only a successful online authentication can re-enable offline authentication. If not specified, this defaults to five (5).

13.2.7. Configuring Services: autofs

About Automount, LDAP, and SSSD

Automount maps are commonly flat files, which define a relationship between a map, a mount directory, and a fileserver. (Automount is described in the [Storage Administration Guide](#).)

For example, let's say that there is a fileserver called **nfs.example.com** which hosts the directory **pub**, and automount is configured to mount directories in the **/shares/** directory. So, the mount location is **/shares/pub**. All of the mounts are listed in the **auto.master** file, which identifies the different mount directories and the files which configure them. The **auto.shares** file then identifies each file server and mount directory which goes into the **/shares/** directory. The relationships could be viewed like this:



Every mount point, then, is defined in two different files (at a minimum): the **auto.master** and **auto.whatever** file, and those files have to be available to each local automount process.

One way for administrators to manage that for large environments is to store the automount configuration in a central LDAP directory, and just configure each local system to point to that LDAP directory. That means that updates only need to be made in a single location, and any new maps are automatically recognized by local systems.

For automount-LDAP configuration, the automount files are stored as LDAP entries, which are then translated into the requisite automount files. Each element is then translated into an LDAP attribute.

The LDAP entries look like this:

```

# container entry
dn: cn=automount,dc=example,dc=com
objectClass: nsContainer
objectClass: top
cn: automount

# master map entry
dn: automountMapName=auto.master,cn=automount,dc=example,dc=com
objectClass: automountMap
objectClass: top
automountMapName: auto.master

# shares map entry
dn: automountMapName=auto.shares,cn=automount,dc=example,dc=com
objectClass: automountMap
objectClass: top
automountMapName: auto.shares

# shares mount point
dn: automountKey=/shares,automountMapName=auto.master,cn=automount,dc=example,dc=com
objectClass: automount
  
```

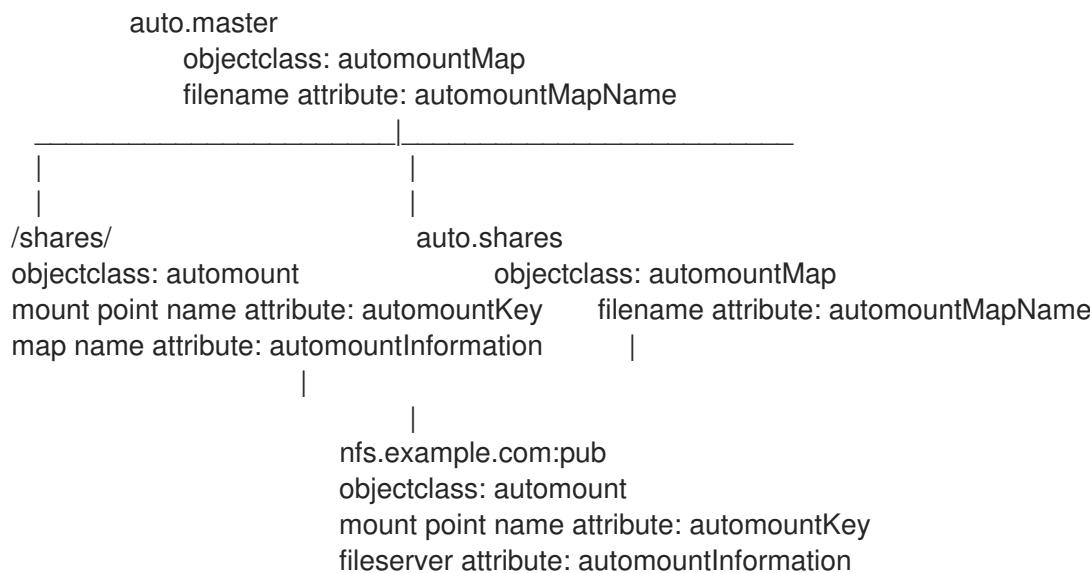
```

objectClass: top
automountKey: /shares
automountInformation: auto.shares

# pub mount point
dn: automountKey=pub,automountMapName=auto.shares,cn=automount,dc=example,dc=com
objectClass: automount
objectClass: top
automountKey: pub
automountInformation: filer.example.com:/pub
description: pub

```

The schema elements, then, match up to the structure like this (with the RFC 2307 schema):



autofs uses those schema elements to derive the automount configuration. The **/etc/sysconfig/autofs** file identifies the LDAP server, directory location, and schema elements used for automount entities:

```

LDAP_URI=ldap://ldap.example.com
SEARCH_BASE="cn=automount,dc=example,dc=com"
MAP_OBJECT_CLASS="automountMap"
ENTRY_OBJECT_CLASS="automount"
MAP_ATTRIBUTE="automountMapName"
ENTRY_ATTRIBUTE="automountKey"
VALUE_ATTRIBUTE="automountInformation"

```

Rather than pointing the automount configuration to the LDAP directory, it can be configured to point to SSSD. SSSD, then, stores all of the information that automount needs, and as a user attempts to mount a directory, that information is cached into SSSD. This offers several advantages for configuration – such as failover, service discovery, and timeouts – as well as performance improvements by reducing the number of connections to the LDAP server. Most important, using SSSD allows all mount information to be cached, so that clients can still successfully mount directories even if the LDAP server goes offline.

Procedure 13.4. Configuring autofs Services in SSSD

1. Make sure that the `autofs` and `sssd-common` packages are installed.
2. Open the `sssd.conf` file.

```
~]# vim /etc/sssd/sssd.conf
```

- Add the **autofs** service to the list of services that SSSD manages.

```
[sssd]
services = nss,pam,autofs
...
```

- Create a new [**autofs**] service configuration section. This section can be left blank; there is only one configurable option, for timeouts for negative cache hits.

This section is required, however, for SSSD to recognize the **autofs** service and supply the default configuration.

```
[autofs]
```

- The automount information is read from a configured LDAP domain in the SSSD configuration, so an LDAP domain must be available. If no additional settings are made, then the configuration defaults to the RFC 2307 schema and the LDAP search base (**ldap_search_base**) for the automount information. This can be customized:

- The directory type, **autofs_provider**; this defaults to the **id_provider** value; a value of *none* explicitly disables autofs for the domain.
- The search base, **ldap_autofs_search_base**.
- The object class to use to recognize map entries, **ldap_autofs_map_object_class**
- The attribute to use to recognize map names, **ldap_autofs_map_name**
- The object class to use to recognize mount point entries, **ldap_autofs_entry_object_class**
- The attribute to use to recognize mount point names, **ldap_autofs_entry_key**
- The attribute to use for additional configuration information for the mount point, **ldap_autofs_entry_value**

For example:

```
[domain/LDAP]
...
autofs_provider=ldap
ldap_autofs_search_base=cn=automount,dc=example,dc=com
ldap_autofs_map_object_class=automountMap
ldap_autofs_entry_object_class=automount
ldap_autofs_map_name=automountMapName
ldap_autofs_entry_key=automountKey
ldap_autofs_entry_value=automountInformation
```

- Save and close the **sssd.conf** file.
- Configure **autofs** to look for the automount map information in SSSD by editing the **nsswitch.conf** file and changing the location from **ldap** to **sss**:

```
# vim /etc/nsswitch.conf
```

```
automount: files sss
```

8. Restart SSSD.

```
# service sssd restart
```

13.2.8. Configuring Services: sudo

About sudo, LDAP, and SSSD

sudo rules are defined in the **sudoers** file, which must be distributed separately to every machine to maintain consistency.

One way for administrators to manage that for large environments is to store the **sudo** configuration in a central LDAP directory, and just configure each local system to point to that LDAP directory. That means that updates only need to be made in a single location, and any new rules are automatically recognized by local systems.

For **sudo**-LDAP configuration, each **sudo** rule is stored as an LDAP entry, with each component of the **sudo** rule defined in an LDAP attribute.

The **sudoers** rule looks like this:

```
Defaults env_keep+=SSH_AUTH_SOCK
...
%wheel    ALL=(ALL)    ALL
```

The LDAP entry looks like this:

```
# sudo defaults
dn: cn=defaults,ou=SUDOers,dc=example,dc=com
objectClass: top
objectClass: sudoRole
cn: defaults
description: Default sudoOptions go here
sudoOption: env_keep+=SSH_AUTH_SOCK

# sudo rule
dn: cn=%wheel,ou=SUDOers,dc=example,dc=com
objectClass: top
objectClass: sudoRole
cn: %wheel
sudoUser: %wheel
sudoHost: ALL
sudoCommand: ALL
```



NOTE

SSSD only caches **sudo** rules which apply to the local system, depending on the value of the **sudoHost** attribute. This can mean that the **sudoHost** value is set to ALL, uses a regular expression that matches the host name, matches the systems netgroup, or matches the systems host name, fully qualified domain name, or IP address.

The **sudo** service can be configured to point to an LDAP server and to pull its rule configuration from those LDAP entries. Rather than pointing the **sudo** configuration to the LDAP directory, it can be configured to point to SSSD. SSSD, then, stores all of the information that **sudo** needs, and every time a user attempts a **sudo**-related operation, the latest **sudo** configuration can be pulled from the LDAP directory (through SSSD). SSSD, however, also caches all of the **sudo** rules, so that users can perform tasks, using that centralized LDAP configuration, even if the LDAP server goes offline.

Procedure 13.5. Configuring sudo with SSSD

All of the SSSD **sudo** configuration options are listed in the **sssd-ldap(5)** man page.

1. Make sure that the `sssd-common` package is installed.

```
~]$ rpm -q sssd-common
```

2. Open the **sssd.conf** file.

```
~]# vim /etc/sssd/sssd.conf
```

3. Add the **sudo** service to the list of services that SSSD manages.

```
[sssd]
services = nss,pam,sudo
....
```

4. Create a new **[sudo]** service configuration section. This section can be left blank; there is only one configurable option, for evaluating the sudo not before/after period.

This section is required, however, for SSSD to recognize the **sudo** service and supply the default configuration.

```
[sudo]
```

5. The **sudo** information is read from a configured LDAP domain in the SSSD configuration, so an LDAP domain must be available. For an LDAP provider, these parameters are required:

- The directory type, **sudo_provider**; this is always **ldap**.
- The search base, **ldap_sudo_search_base**.
- The URI for the LDAP server, **ldap_uri**.

For example:

```
[domain/LDAP]
id_provider = ldap

sudo_provider = ldap
ldap_uri = ldap://example.com
ldap_sudo_search_base = ou=sudoers,dc=example,dc=com
```

For an Identity Management (IdM or IPA) provider, there are additional parameters required to perform Kerberos authentication when connecting to the server.

```
[domain/IDM]
id_provider = ipa
ipa_domain = example.com
ipa_server = ipa.example.com
ldap_tls_cacert = /etc/ipa/ca.crt

sudo_provider = ldap
ldap_uri = ldap://ipa.example.com
ldap_sudo_search_base = ou=sudoers,dc=example,dc=com
ldap_sasl_mech = GSSAPI
ldap_sasl_authid = host/hostname.example.com
ldap_sasl_realm = EXAMPLE.COM
krb5_server = ipa.example.com
```



NOTE

The **sudo_provider** type for an Identity Management provider is still **ldap**.

- Set the intervals to use to refresh the **sudo** rule cache.

The cache for a specific system user is always checked and updated whenever that user performs a task. However, SSSD caches all rules which relate to the local system. That complete cache is updated in two ways:

- Incrementally, meaning only changes to rules since the last full update (**ldap_sudo_smart_refresh_interval**, the time in seconds); the default is 15 minutes,
- Fully, which dumps the entire caches and pulls in all of the current rules on the LDAP server(**ldap_sudo_full_refresh_interval**, the time in seconds); the default is six hours.

These two refresh intervals are set separately. For example:

```
[domain/LDAP]
...
ldap_sudo_full_refresh_interval=86400
ldap_sudo_smart_refresh_interval=3600
```



NOTE

SSSD only caches **sudo** rules which apply to the local system. This can mean that the **sudoHost** value is set to ALL, uses a regular expression that matches the host name, matches the systems netgroup, or matches the systems host name, fully qualified domain name, or IP address.

- Optionally, set any values to change the schema used for **sudo** rules.

Schema elements are set in the **ldap_sudorule_*** attributes. By default, all of the schema elements use the schema defined in [sudoers.ldap](#); these defaults will be used in almost all deployments.

- Save and close the **sssd.conf** file.

- Configure **sudo** to look for rules configuration in SSSD by editing the **nsswitch.conf** file and adding the **sss** location:

```
~]# vim /etc/nsswitch.conf
sudoers: files sss
```

- Restart SSSD.

```
~]# service sssd restart
```

13.2.9. Configuring Services: OpenSSH and Cached Keys

OpenSSH creates secure, encrypted connections between two systems. One machine authenticates to another machine to allow access; the authentication can be of the machine itself for server connections or of a user on that machine. OpenSSH is described in more detail in [Chapter 14, OpenSSH](#).

This authentication is performed through *public-private key pairs* that identify the authenticating user or machine. The remote machine or user attempting to access the machine presents a key pair. The local machine then elects whether to trust that remote entity; if it is trusted, the public key for that remote machine is stored in the **known_hosts** file or for the remote user in **authorized_keys**. Whenever that remote machine or user attempts to authenticate again, the local system checks the **known_hosts** or **authorized_keys** file first to see if that remote entity is recognized and trusted. If it is, then access is granted.

The first problem comes in verifying those identities reliably.

The **known_hosts** file is a triplet of the machine name, its IP address, and its public key:

```
server.example.com,255.255.255.255 ssh-rsa
AbcdEfg1234ZYX098776/AbcdEfg1234ZYX098776/AbcdEfg1234ZYX098776=
```

The **known_hosts** file can quickly become outdated for a number of different reasons: systems using DHCP cycle through IP addresses, new keys can be re-issued periodically, or virtual machines or services can be brought online and removed. This changes the host name, IP address, and key triplet.

Administrators have to clean and maintain a current **known_hosts** file to maintain security. (Or system users get in the habit of accepting any machine and key presented, which negates the security benefits of key-based security.)

Additionally, a problem for both machines and users is distributing keys in a scalable way. Machines can send their keys as part of establishing an encrypted session, but users have to supply their keys in advance. Simply propagating and then updating keys consistently is a difficult administrative task.

Lastly, SSH key and machine information are only maintained locally. There may be machines or users on the network which are recognized and trusted by some systems and not by others because the **known_hosts** file has not been updated uniformly.

The goal of SSSD is to serve as a credentials cache. This includes working as a credentials cache for SSH public keys for machines and users. OpenSSH is configured to reference SSSD to check for cached keys; SSSD uses Red Hat Linux's Identity Management (IPA) domain as an identity, and Identity Management actually stores the public keys and host information.

**NOTE**

Only Linux machines enrolled, or joined, in the Identity Management domain can use SSSD as a key cache for OpenSSH. Other Unix machines and Windows machines must use the regular authentication mechanisms with the **known_hosts** file.

Configuring OpenSSH to Use SSSD for Host Keys

OpenSSH is configured in either a user-specific configuration file (`~/.ssh/config`) or a system-wide configuration file (`/etc/ssh/ssh_config`). The user file has precedence over the system settings and the first obtained value for a parameter is used. The formatting and conventions for this file are covered in [Chapter 14, OpenSSH](#).

In order to manage host keys, SSSD has a tool, **sss_ssh_knownhostsproxy**, which performs two operations:

1. Asks SSSD to retrieve the public host key from the Identity Management server and store it in the `/var/lib/sss/pubconf/known_hosts` file.
2. Establishes a connection with the host machine, using either a socket (the default) or a proxy command.

This tool has the format:

```
sss_ssh_knownhostsproxy [-d sssd_domain] [-p ssh_port] HOST [PROXY_COMMAND]
```

Table 13.4. `sss_ssh_knownhostsproxy` Options

Short Argument	Long Argument	Description
	HOSTNAME	Gives the host name of the host to check and connect to. In the OpenSSH configuration file, this can be a token, %h .
	PROXY_COMMAND	Passes a proxy command to use to connect to the SSH client. This is similar to running <code>ssh -o ProxyCommand=value</code> . This option is used when running sss_ssh_knownhostsproxy from the command line or through another script, but is not necessary in the OpenSSH configuration file.
-d sssd_domain	--domain sssd_domain	Only searches for public keys in entries in the specified domain. If not given, SSSD searches for keys in all configured domains.
-p port	--port port	Uses this port to connect to the SSH client. By default, this is port 22.

To use this SSSD tool, add or edit two parameters to the **ssh_config** or `~/.ssh/config` file:

- Specify the command to use to connect to the SSH client (**ProxyCommand**). This is the **sss_ssh_knownhostsproxy**, with the desired arguments and host name.
- Specify the location of the SSSD hosts file (**GlobalKnownHostsFile**).

For example, this looks for public keys in all configured SSSD domains and connects over whatever port and host are supplied:

```
ProxyCommand /usr/bin/ssss_ssh_knownhostsproxy -p %p %h
GlobalKnownHostsFile /var/lib/ssss/pubconf/known_hosts
```

Configuring OpenSSH to Use SSSD for User Keys

SSSD can provide user public keys to OpenSSH. The keys are read by the SSH daemon, **sshd**, directly from the output of the **ssss_authorizedkeys** tool and are not stored in a file.

To configure **sshd** to read a user's public keys from an external program, in this case the **ssss_authorizedkeys** tool, use the **AuthorizedKeysCommand** directive in the **/etc/ssh/sshd_config** file.

The **ssss_authorizedkeys** tool can be used to acquire SSH public keys from the user entries in the Identity Management (IPA) domain and output them in OpenSSH **authorized_keys** format. The command has the following format:

```
ssss_authorizedkeys [-d sssd_domain] USER
```

Table 13.5. sss_authorizedkeys Options

Short Argument	Long Argument	Description
	<i>USER</i>	The user name or account name for which to obtain the public key. In the OpenSSH configuration file, this can be represented by a token, %u .
-d <i>sssd_domain</i>	--domain <i>sssd_domain</i>	Only search for public keys in entries in the specified domain. If not given, SSSD searches for keys in all configured domains.

This feature is configured in **/etc/ssh/sshd_config** as follows:

```
AuthorizedKeysCommand /usr/bin/ssss_authorizedkeys
AuthorizedKeysCommandRunAs nobody
```

These and further options are documented in the **sshd_config(5)** man page. Note that the **sshd** service must be restarted for any changes to take effect.

13.2.10. SSSD and Identity Providers (Domains)

SSSD recognizes *domains*, which are entries within the SSSD configuration file associated with different, external data sources. Domains are a combination of an identity provider (for user information) and, optionally, other providers such as authentication (for authentication requests) and for other operations, such as password changes. (The identity provider can also be used for all operations, if all operations are performed within a single domain or server.)

SSSD works with different LDAP identity providers (including OpenLDAP, Red Hat Directory Server, and Microsoft Active Directory) and can use native LDAP authentication, Kerberos authentication, or provider-specific authentication protocols (such as Active Directory).

A domain configuration defines the *identity provider*, the *authentication provider*, and any specific configuration to access the information in those providers. There are several types of identity and authentication providers:

- LDAP, for general LDAP servers

- Active Directory (an extension of the LDAP provider type)
- Identity Management (an extension of the LDAP provider type)
- Local, for the local SSSD database
- Proxy
- Kerberos (authentication provider only)

The identity and authentication providers can be configured in different combinations in the domain entry. The possible combinations are listed in [Table 13.6, “Identity Store and Authentication Type Combinations”](#).

Table 13.6. Identity Store and Authentication Type Combinations

Identification Provider	Authentication Provider
Identity Management (LDAP)	Identity Management (LDAP)
Active Directory (LDAP)	Active Directory (LDAP)
Active Directory (LDAP)	Kerberos
LDAP	LDAP
LDAP	Kerberos
proxy	LDAP
proxy	Kerberos
proxy	proxy

Along with the domain entry itself, the domain name must be added to the list of domains that SSSD will query. For example:

```
[sssd]
domains = LOCAL,Name
...
[domain/Name]
id_provider = type
auth_provider = type
provider_specific = value
global = value
```

global attributes are available to any type of domain, such as cache and time out settings. Each identity and authentication provider has its own set of required and optional configuration parameters.

Table 13.7. General [domain] Configuration Parameters

Parameter	Value Format	Description
id_provider	string	<p>Specifies the data back end to use for this domain. The supported identity back ends are:</p> <ul style="list-style-type: none"> ● ldap ● ipa (Identity Management in Red Hat Enterprise Linux) ● ad (Microsoft Active Directory) ● proxy, for a legacy NSS provider, such as nss_nis. Using a proxy ID provider also requires specifying the legacy NSS library to load to start successfully, set in the proxy_lib_name option. ● local, the SSSD internal local provider
auth_provider	string	<p>Sets the authentication provider used for the domain. The default value for this option is the value of id_provider. The supported authentication providers are ldap, ipa, ad, krb5 (Kerberos), proxy, and none.</p>
min_id,max_id	integer	<p><i>Optional.</i> Specifies the UID and GID range for the domain. If a domain contains entries that are outside that range, they are ignored. The default value for min_id is 1; the default value for max_id is 0, which is unlimited.</p>
		 <p>IMPORTANT</p> <p>The default min_id value is the same for all types of identity provider. If LDAP directories are using UID numbers that start at one, it could cause conflicts with users in the local /etc/passwd file. To avoid these conflicts, set min_id to 1000 or higher as possible.</p>
cache_credentials	Boolean	<p><i>Optional.</i> Specifies whether to store user credentials in the local SSSD domain database cache. The default value for this parameter is false. Set this value to true for domains other than the LOCAL domain to enable offline authentication.</p>
entry_cache_timeout	integer	<p><i>Optional.</i> Specifies how long, in seconds, SSSD should cache positive cache hits. A positive cache hit is a successful query.</p>

Parameter	Value Format	Description
use_fully_qualified_names	Boolean	<p><i>Optional.</i> Specifies whether requests to this domain require fully qualified domain names. If set to true, all requests to this domain must use fully qualified domain names. It also means that the output from the request displays the fully-qualified name. Restricting requests to fully qualified user names allows SSSD to differentiate between domains with users with conflicting user names.</p> <p>If use_fully_qualified_names is set to false, it is possible to use the fully-qualified name in the requests, but only the simplified version is displayed in the output.</p> <p>SSSD can only parse names based on the domain name, not the realm name. The same name can be used for both domains and realms, however.</p>

13.2.11. Creating Domains: LDAP

An LDAP domain means that SSSD uses an LDAP directory as the identity provider (and, optionally, also as an authentication provider). SSSD supports several major directory services:

- Red Hat Directory Server
- OpenLDAP
- Identity Management (IdM or IPA)
- Microsoft Active Directory 2008 R2



NOTE

All of the parameters available to a general LDAP identity provider are also available to Identity Management and Active Directory identity providers, which are subsets of the LDAP provider.

Parameters for Configuring an LDAP Domain

An LDAP directory can function as both an identity provider and an authentication provider. The configuration requires enough information to identify and connect to the user directory in the LDAP server, but the way that those connection parameters are defined is flexible.

Other options are available to provide more fine-grained control, like specifying a user account to use to connect to the LDAP server or using different LDAP servers for password operations. The most common options are listed in [Table 13.8, “LDAP Domain Configuration Parameters”](#).



NOTE

Server-side password policies always take precedence over the policy enabled from the client side. For example, when setting the **ldap_pwd_policy=shadow** option, the policies defined with the **shadow** LPAD attributes for a user have no effect on whether the password policy is enabled on the OpenLDAP server.

**NOTE**

Many other options are listed in the man page for LDAP domain configuration, **sssd-ldap(5)**.

Table 13.8. LDAP Domain Configuration Parameters

Parameter	Description
ldap_uri	Gives a comma-separated list of the URIs of the LDAP servers to which SSSD will connect. The list is given in order of preference, so the first server in the list is tried first. Listing additional servers provides failover protection. This can be detected from the DNS SRV records if it is not given.
ldap_search_base	<p>Gives the base DN to use for performing LDAP user operations.</p> <p>IMPORTANT</p> <p>If used incorrectly, ldap_search_base might cause SSSD lookups to fail.</p> <p>With an AD provider, setting ldap_search_base is not required. The AD provider automatically discovers all the necessary information. Red Hat recommends not to set the parameter in this situation and instead rely on what the AD provider discovers.</p>
ldap_tls_reqcert	<p>Specifies how to check for SSL server certificates in a TLS session. There are four options:</p> <ul style="list-style-type: none"> ● <i>never</i> disables requests for certificates. ● <i>allow</i> requests a certificate, but proceeds normally even if no certificate is given or a bad certificate is given. ● <i>try</i> requests a certificate and proceeds normally if no certificate is given. If a bad certificate is given, the session terminates. ● <i>demand</i> and <i>hard</i> are the same option. This requires a valid certificate or the session is terminated. <p>The default is <i>hard</i>.</p>
ldap_tls_cacert	<p>Gives the full path and file name to the file that contains the CA certificates for all of the CAs that SSSD recognizes. SSSD will accept any certificate issued by these CAs.</p> <p>This uses the OpenLDAP system defaults if it is not given explicitly.</p>
ldap_referrals	<p>Sets whether SSSD will use LDAP referrals, meaning forwarding queries from one LDAP database to another. SSSD supports database-level and subtree referrals. For referrals within the same LDAP server, SSSD will adjust the DN of the entry being queried. For referrals that go to different LDAP servers, SSSD does an exact match on the DN. Setting this value to true enables referrals; this is the default.</p> <p>Referrals can negatively impact overall performance because of the time spent attempting to trace referrals. Disabling referral checking can significantly improve performance.</p>

Parameter	Description
ldap_schema	<p>Sets what version of schema to use when searching for user entries. This can be rfc2307, rfc2307bis, ad, or ipa. The default is rfc2307.</p> <p>In RFC 2307, group objects use a multi-valued attribute, memberuid, which lists the names of the users that belong to that group. In RFC 2307bis, group objects use the member attribute, which contains the full distinguished name (DN) of a user or group entry. RFC 2307bis allows nested groups using the member attribute. Because these different schema use different definitions for group membership, using the wrong LDAP schema with SSSD can affect both viewing and managing network resources, even if the appropriate permissions are in place.</p> <p>For example, with RFC 2307bis, all groups are returned when using nested groups or primary/secondary groups.</p> <pre>\$ id uid=500(myserver) gid=500(myserver) groups=500(myserver),510(myothergroup)</pre> <p>If SSSD is using RFC 2307 schema, only the primary group is returned.</p> <p>This setting only affects how SSSD determines the group members. It does not change the actual user data.</p>
ldap_search_timeout	<p>Sets the time, in seconds, that LDAP searches are allowed to run before they are canceled and cached results are returned.</p> <p>When an LDAP search times out, SSSD automatically switches to offline mode.</p>
ldap_network_timeout	<p>Sets the time, in seconds, SSSD attempts to poll an LDAP server after a connection attempt fails. The default is six seconds.</p>
ldap_opt_timeout	<p>Sets the time, in seconds, to wait before aborting synchronous LDAP operations if no response is received from the server. This option also controls the timeout when communicating with the KDC in case of a SASL bind. The default is five seconds.</p>

LDAP Domain Example

The LDAP configuration is very flexible, depending on your specific environment and the SSSD behavior. These are some common examples of an LDAP domain, but the SSSD configuration is not limited to these examples.

**NOTE**

Along with creating the domain entry, add the new domain to the list of domains for SSSD to query in the **sssd.conf** file. For example:

```
domains = LOCAL,LDAP1,AD,PROXYNIS
```

Example 13.2. A Basic LDAP Domain Configuration

An LDAP domain requires three things:

- An LDAP server
- The search base
- A way to establish a secure connection

The last item depends on the LDAP environment. SSSD requires a secure connection since it handles sensitive information. This connection can be a dedicated TLS/SSL connection or it can use Start TLS.

Using a dedicated TLS/SSL connection uses an LDAPS connection to connect to the server and is therefore set as part of the **ldap_uri** option:

```
# An LDAP domain
[domain/LDAP]
cache_credentials = true

id_provider = ldap
auth_provider = ldap

ldap_uri = ldaps://ldap.example.com:636
ldap_search_base = dc=example,dc=com
```

Using Start TLS requires a way to input the certificate information to establish a secure connection dynamically over an insecure port. This is done using the **ldap_id_use_start_tls** option to use Start TLS and then **ldap_tls_cacert** to identify the CA certificate which issued the SSL server certificates.

```
# An LDAP domain
[domain/LDAP]
cache_credentials = true

id_provider = ldap
auth_provider = ldap

ldap_uri = ldap://ldap.example.com
ldap_search_base = dc=example,dc=com
ldap_id_use_start_tls = true
ldap_tls_reqcert = demand
ldap_tls_cacert = /etc/pki/tls/certs/ca-bundle.crt
```

13.2.12. Creating Domains: Identity Management (IdM)

The Identity Management (IdM or IPA) identity provider is an extension of a generic LDAP provider. All of the configuration options for an LDAP provider are available to the IdM provider, as well as some additional parameters which allow SSSD to work as a client of the IdM domain and extend IdM functionality.

Identity Management can work as a provider for identities, authentication, access control rules, and passwords, all of the `*_provider` parameters for a domain. Additionally, Identity Management has configuration options within its own domain to manage SELinux policies, automount information, and host-based access control. All of those features in IdM domains can be tied to SSSD configuraiton, allowing those security-related policies to be applied and cached for system users.

Example 13.3. Basic IdM Provider

An IdM provider, like an LDAP provider, can be set to serve several different services, including identity, authentication, and access control

For IdM servers, there are two additional settings which are very useful (although not required):

- Use the specific IdM schema rather than the default RFC 2307 schema.
- Set SSSD to update the Identity Management domain's DNS server with the IP address of this client when the client first connects to the IdM domain.

```
[sssd]
domains = local.example.com
...
[domain/example.com]
id_provider = ipa
ipa_server = ipaserver.example.com
ipa_hostname = ipa1.example.com
auth_provider = ipa
access_provider = ipa
chpass_provider = ipa

# set which schema to use
ldap_schema = ipa

# automatically update IdM DNS records
ipa_dyndns_update = true
```

Identity Management defines and maintains security policies and identities for users across a Linux domain. This includes access control policies, SELinux policies, and other rules. Some of these elements in the IdM domain interact directly with SSSD, using SSSD as an IdM client – and those features can be managed in the IdM domain entry in `sssd.conf`.

Most of the configuration parameters relate to setting schema elements (which is not relevant in most deployments because IdM uses a fixed schema) and never need to be changed. In fact, none of the features in IdM require client-side settings. But there may be circumstances where tweaking the behavior is helpful.

Example 13.4. IdM Provider with SELinux

IdM can define SELinux user policies for system users, so it can work as an SELinux provider for

SSSD. This is set in the **selinux_provider** parameter. The provider defaults to the **id_provider** value, so this is not necessary to set explicitly to support SELinux rules. However, it can be useful to explicitly *disable* SELinux support for the IdM provider in SSSD.

```
selinux_provider = ipa
```

Example 13.5. IdM Provider with Host-Based Access Control

IdM can define host-based access controls, restricting access to services or entire systems based on what host a user is using to connect or attempting to connect to. These rules can be evaluated and enforced by SSSD as part of the access provider behavior.

For host-based access controls to be in effect, the Identity Management server must be the access provider, at a minimum.

There are two options which can be set for how SSSD evaluates host-based access control rules:

- SSSD can evaluate what machine (source host) the user is using to connect to the IdM resource; this is disabled by default, so that only the target host part of the rule is evaluated.
- SSSD can refresh the host-based access control rules in its cache at a specified interval.

For example:

```
access_provider = ipa
ipa_hbac_refresh = 120

# check for source machine rules; disabled by default
ipa_hbac_support_srchost = true
```

Example 13.6. Identity Management with Cross-Realm Kerberos Trusts

Identity Management (IdM or IPA) can be configured with trusted relationships between Active Directory DNS domains and Kerberos realms. This allows Active Directory users to access services and hosts on Linux systems.

There are two configuration settings in SSSD that are used with cross-realm trusts:

- A service that adds required data to Kerberos tickets
- A setting to support subdomains

Kerberos Ticket Data

Microsoft uses a special authorization structure called *privileged access certificates* or MS-PAC. A PAC is embedded in a Kerberos ticket as a way of identifying the entity to other Windows clients and servers in the Windows domain.

SSSD has a special PAC service which generates the additional data for Kerberos tickets. When using an Active Directory domain, it may be necessary to include the PAC data for Windows users. In that case, enable the **pac** service in SSSD:

```
[sssd]
services = nss, pam, pac
...
```

Windows Subdomains

Normally, a domain entry in SSSD corresponds directly to a single identity provider. However, with IdM cross-realm trusts, the IdM domain can trust another domain, so that the domains are transparent to each other. SSSD can follow that trusted relationship, so that if an IdM domain is configured, any Windows domain is also automatically searched and supported by SSSD – without having to be configured in a domain section in SSSD.

This is configured by adding the **subdomains_provider** parameter to the IdM domain section. This is actually an optional parameter; if a subdomain is discovered, then SSSD defaults to using the **ipa** provider type. However, this parameter can also be used to disable subdomain fetches by setting a value of **none**.

```
[domain/IDM]
...
subdomains_provider = ipa
get_domains_timeout = 300
```

13.2.13. Creating Domains: Active Directory

The Active Directory identity provider is an extension of a generic LDAP provider. All of the configuration options for an LDAP provider are available to the Active Directory provider, as well as some additional parameters related to user accounts and identity mapping between Active Directory and system users.

There are some fundamental differences between standard LDAP servers and an Active Directory server. When configuring an Active Directory provider, there are some configuration areas, then, which require specific configuration:

- Identities using a Windows security ID must be mapped to the corresponding Linux system user ID.
- Searches must account for the range retrieval extension.
- There may be performance issues with LDAP referrals.

Mapping Active Directory Security IDs and Linux User IDs

There are inherent structural differences between how Windows and Linux handle system users and in the user schemas used in Active Directory and standard LDAPv3 directory services. When using an Active Directory identity provider with SSSD to manage system users, it is necessary to reconcile the Active Directory-style user to the new SSSD user. There are two ways to do this:

- Using Services for Unix to insert POSIX attributes on Windows user and group entries, and then having those attributes pulled into PAM/NSS
- Using ID mapping on SSSD to create a map between Active Directory security IDs (SIDs) and the generated UIDs on Linux

ID mapping is the simplest option for most environments because it requires no additional packages or configuration on Active Directory.

The Mechanism of ID Mapping

Linux/Unix systems use a local user ID number and group ID number to identify users on the system. These UID:GID numbers are a simple integer, such as 501:501. These numbers are simple because they are always created and administered locally, even for systems which are part of a larger Linux/Unix domain.

Microsoft Windows and Active Directory use a different user ID structure to identify users, groups, and machines. Each ID is constructed of different segments that identify the security version, the issuing authority type, the machine, and the identity itself. For example:

S-1-5-21-3623811015-3361044348-30300820-1013

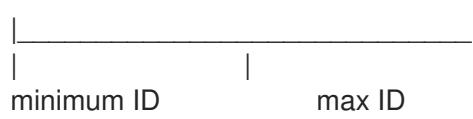
The third through sixth blocks are the machine identifier:

S-1-5-21-3623811015-3361044348-30300820-1013

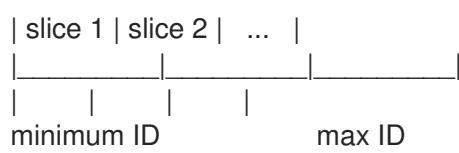
The last block is the *relative identifier* (RID) which identifies the specific entity:

S-1-5-21-3623811015-3361044348-30300820-1013

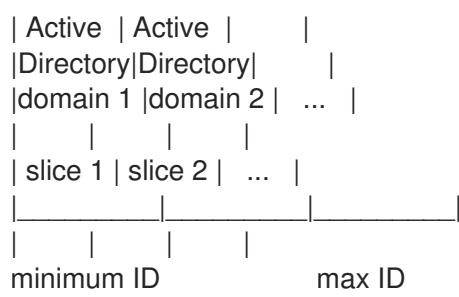
A range of possible ID numbers are always assigned to SSSD. (This is a local range, so it is the same for every machine.)



This range is divided into 10,000 sections (by default), with each section allocated 200,000 IDs.



When a new Active Directory domain is detected, the SID is hashed. Then, SSSD takes the modulus of the hash and the number of available sections to determine which ID section to assign to the Active Directory domain. This is a reliably consistent means of assigning ID sections, so the same ID range is assigned to the same Active Directory domain on most client machines.



**NOTE**

While the method of assigning ID sections is consistent, **ID mapping is based on the order that an Active Directory domain is encountered on a client machine** – so it may not result in consistent ID range assignments on all Linux client machines. If consistency is required, then consider disabling ID mapping and using explicit POSIX attributes.

ID Mapping Parameters

ID mapping is enabled in two parameters, one to enable the mapping and one to load the appropriate Active Directory user schema:

```
ldap_id_mapping = True
ldap_schema = ad
```

**NOTE**

When ID mapping is enabled, the ***uidNumber*** and ***gidNumber*** attributes are ignored. This prevents any manually-assigned values. If *any* values must be manually assigned, then *all* values must be manually assigned, and ID mapping should be disabled.

Mapping Users

When an Active Directory user attempts to log into a local system service for the first time, an entry for that user is created in the SSSD cache. The remote user is set up much like a system user:

- A system UID is created for the user based on his SID and the ID range for that domain.
- A GID is created for the user, which is identical to the UID.
- A private group is created for the user.
- A home directory is created, based on the home directory format in the **sssd.conf** file.
- A shell is created, according to the system defaults or the setting in the **sssd.conf** file.
- If the user belongs to any groups in the Active Directory domain, then, using the SID, SSSD adds the user to those groups on the Linux system.

Active Directory Users and Range Retrieval Searches

Microsoft Active Directory has an attribute, ***MaxValRange***, which sets a limit on how many values for a multi-valued attribute will be returned. This is the *range retrieval* search extension. Essentially, this runs multiple mini-searches, each returning a subset of the results within a given range, until all matches are returned.

For example, when doing a search for the ***member*** attribute, each entry could have multiple values, and there can be multiple entries with that attribute. If there are 2000 matching results (or more), then ***MaxValRange*** limits how many are displayed at once; this is the value range. The given attribute then has an additional flag set, showing which range in the set the result is in:

```
attribute:range=low-high:value
```

For example, results 100 to 500 in a search:

```
member;range=99-499: cn=John Smith...
```

This is described in the Microsoft documentation at <http://msdn.microsoft.com/en-us/library/cc223242.aspx>.

SSSD supports range retrievals with Active Directory providers as part of user and group management, without any additional configuration.

However, some LDAP provider attributes which are available to configure searches – such as **ldap_user_search_base** – are not performant with range retrievals. Be cautious when configuring search bases in the Active Directory provider domain and consider what searches may trigger a range retrieval.

Performance and LDAP Referrals

Referrals can negatively impact overall performance because of the time spent attempting to trace referrals. There is particularly bad performance degradation when referral chasing is used with an Active Directory identity provider. Disabling referral checking can significantly improve performance.

LDAP referrals are enabled by default, so they must be explicitly disabled in the LDAP domain configuration. For example:

```
ldap_referrals = false
```

Active Directory as Other Provider Types

Active Directory can be used as an identity provider and as an access, password, and authentication provider.

There are a number of options in the generic LDAP provider configuration which can be used to configure an Active Directory provider. Using the **ad** value is a short-cut which automatically pulls in the parameters and values to configure a given provider for Active Directory. For example, using **access_provider = ad** to configure an Active Directory access provider expands to this configuration using the explicit LDAP provider parameters:

```
access_provider = ldap
ldap_access_order = expire
ldap_account_expire_policy = ad
```

Procedure 13.6. Configuring an Active Directory Identity Provider

Active Directory can work as a provider for identities, authentication, access control rules, and passwords, all of the ***_provider** parameters for a domain. Additionally, it is possible to load the native Active Directory schema for user and group entries, rather than using the default RFC 2307.

1. Make sure that both the Active Directory and Linux systems have a properly configured environment.
 - o Name resolution must be properly configured, particularly if service discovery is used with SSSD.
 - o The clocks on both systems must be in sync for Kerberos to work properly.
2. Set up the Linux system as an Active Directory client and enroll it within the Active Directory domain. This is done by configuring the Kerberos and Samba services on the Linux system.
 - a. Set up Kerberos to use the Active Directory Kerberos realm.
 - i. Open the Kerberos client configuration file.

```
~]# vim /etc/krb5.conf
```

- ii. Configure the **[logging]** and **[libdefaults]** sections so that they connect to the Active Directory realm.

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

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

If autodiscovery is not used with SSSD, then also configure the **[realms]** and **[domain_realm]** sections to explicitly define the Active Directory server.

- b. Configure the Samba server to connect to the Active directory server.

- i. Open the Samba configuration file.

```
~]# vim /etc/samba/smb.conf
```

- ii. Set the Active Directory domain information in the **[global]** section.

```
[global]
workgroup = EXAMPLE
client signing = yes
client use spnego = yes
kerberos method = secrets and keytab
log file = /var/log/samba/%m.log
password server = AD.EXAMPLE.COM
realm = EXAMPLE.COM
security = ads
```

- c. Add the Linux machine to the Active Directory domain.

- i. Obtain Kerberos credentials for a Windows administrative user.

```
~]# kinit Administrator
```

- ii. Add the machine to the domain using the **net** command.

```
~]# net ads join -k
Joined 'server' to dns domain 'example.com'
```

This creates a new keytab file, **/etc/krb5.keytab**.

List the keys for the system and check that the host principal is there.

```
~]# klist -k
```

3. Use **authconfig** to enable SSSD for system authentication.

```
# authconfig --update --enablesssd --enablessdauth
```

4. Set the Active Directory domain as an identity provider in the SSSD configuration, as shown in [Example 13.7, "An Active Directory 2008 R2 Domain"](#) and [Example 13.8, "An Active Directory 2008 R2 Domain with ID Mapping"](#).

5. Restart the SSH service to load the new PAM configuration.

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

6. Restart SSSD after changing the configuration file.

```
~]# service sssd restart
```

Example 13.7. An Active Directory 2008 R2 Domain

```
~]# vim /etc/sssd/sssd.conf

[sssd]
config_file_version = 2
domains = ad.example.com
services = nss, pam

...
[domain/ad.example.com]
id_provider = ad
ad_server = ad.example.com
ad_hostname = ad.example.com
auth_provider = ad
chpass_provider = ad
access_provider = ad

# defines user/group schema type
ldap_schema = ad

# using explicit POSIX attributes in the Windows entries
ldap_id_mapping = False

# caching credentials
cache_credentials = true

# access controls
ldap_access_order = expire
ldap_account_expire_policy = ad
ldap_force_upper_case_realm = true

# performance
ldap_referrals = false
```

There are two parameters that are critical for ID mapping: the Active Directory schema must be loaded (**ldap_schema**) and ID mapping must be explicitly enabled (**ldap_id_mapping**).

Example 13.8. An Active Directory 2008 R2 Domain with ID Mapping

```
~]# vim /etc/sssd/sssd.conf

[sssd]
config_file_version = 2
domains = ad.example.com
services = nss, pam

...
[domain/ad.example.com]
id_provider = ad
ad_server = ad.example.com
ad_hostname = ad.example.com
auth_provider = ad
chpass_provider = ad
access_provider = ad

# defines user/group schema type
ldap_schema = ad

# for SID-UID mapping
ldap_id_mapping = True

# caching credentials
cache_credentials = true

# access controls
ldap_access_order = expire
ldap_account_expire_policy = ad
ldap_force_upper_case_realm = true

# performance
ldap_referrals = false
```

All of the potential configuration attributes for an Active Directory domain are listed in the **sssd-ldap(5)** and **sssd-ad(5)** man pages.

13.2.14. Configuring Domains: Active Directory as an LDAP Provider (Alternative)

While Active Directory can be configured as a type-specific identity provider, it can also be configured as a pure LDAP provider with a Kerberos authentication provider.

Procedure 13.7. Configuring Active Directory as an LDAP Provider

1. It is recommended that SSSD connect to the Active Directory server using SASL, which means that the local host must have a service keytab *for the Windows domain* on the Linux host.

This keytab can be created using Samba.

- Configure the **/etc/krb5.conf** file to use the Active Directory realm.

```
[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
```

- Set the Samba configuration file, **/etc/samba/smb.conf**, to point to the Windows Kerberos realm.

```
[global]
workgroup = EXAMPLE
client signing = yes
client use spnego = yes
kerberos method = secrets and keytab
log file = /var/log/samba/%m.log
password server = AD.EXAMPLE.COM
realm = EXAMPLE.COM
security = ads
```

- To initialize Kerberos, type the following command as **root**:

```
~]# kinit Administrator@EXAMPLE.COM
```

- Then, run the **net ads** command to log in as an administrator principal. This administrator account must have sufficient rights to add a machine to the Windows domain, but it does not require domain administrator privileges.

```
~]# net ads join -U Administrator
```

- Run **net ads** again to add the host machine to the domain. This can be done with the host principal (*host/FQDN*) or, optionally, with the NFS service (*nfs/FQDN*).

```
~]# net ads join createupn="host/rhel-server.example.com@AD.EXAMPLE.COM" -U  
Administrator
```

2. Make sure that the Services for Unix package is installed on the Windows server.
3. Set up the Windows domain which will be used with SSSD.
 - a. On the Windows machine, open **Server Manager**.
 - b. Create the Active Directory Domain Services role.
 - c. Create a new domain, such as **ad.example.com**.
 - d. Add the Identity Management for UNIX service to the Active Directory Domain Services role. Use the Unix NIS domain as the domain name in the configuration.
4. On the Active Directory server, create a group for the Linux users.
 - a. Open **Administrative Tools** and select **Active Directory Users and Computers**.
 - b. Select the Active Directory domain, **ad.example.com**.
 - c. In the **Users** tab, right-click and select **Create a New Group**.
 - d. Name the new group **unixusers**, and save.
 - e. Double-click the **unixusers** group entry, and open the **Users** tab.
 - f. Open the **Unix Attributes** tab.
 - g. Set the NIS domain to the NIS domain that was configured for **ad.example.com** and, optionally, set a group ID (GID) number.
5. Configure a user to be part of the Unix group.
 - a. Open **Administrative Tools** and select **Active Directory Users and Computers**.
 - b. Select the Active Directory domain, **ad.example.com**.
 - c. In the **Users** tab, right-click and select **Create a New User**.
 - d. Name the new user **aduser**, and make sure that the **User must change password at next logon** and **Lock account** check boxes are *not* selected.

Then save the user.

- e. Double-click the **aduser** user entry, and open the **Unix Attributes** tab. Make sure that the Unix configuration matches that of the Active Directory domain and the **unixgroup** group:
 - The NIS domain, as created for the Active Directory domain
 - The UID
 - The login shell, to **/bin/bash**
 - The home directory, to **/home/aduser**

- The primary group name, to **unixusers**



NOTE

Password lookups on large directories can take several seconds per request. The initial user lookup is a call to the LDAP server. Unindexed searches are much more resource-intensive, and therefore take longer, than indexed searches because the server checks every entry in the directory for a match. To speed up user lookups, index the attributes that are searched for by SSSD:

- uid
- uidNumber
- gidNumber
- gecos

6. On the Linux system, configure the SSSD domain.

```
~]# vim /etc/sssd/sssd.conf
```

For a complete list of LDAP provider parameters, see the **sssd-ldap(5)** man pages.

Example 13.9. An Active Directory 2008 R2 Domain with Services for Unix

```
[sssd]
config_file_version = 2
domains = ad.example.com
services = nss, pam

...
[domain/ad.example.com]
cache_credentials = true

# for performance
ldap_referrals = false

id_provider = ldap
auth_provider = krb5
chpass_provider = krb5
access_provider = ldap

ldap_schema = rfc2307bis

ldap_sasl_mech = GSSAPI
ldap_sasl_authid = host/rhel-server.example.com@AD.EXAMPLE.COM

#provide the schema for services for unix
ldap_schema = rfc2307bis

ldap_user_search_base = ou=user accounts,dc=ad,dc=example,dc=com
ldap_user_object_class = user
ldap_user_home_directory = unixHomeDirectory
```

```

ldap_user_principal = userPrincipalName

# optional - set schema mapping
# parameters are listed in sssd-ldap
ldap_user_object_class = user
ldap_user_name = sAMAccountName

ldap_group_search_base = ou=groups,dc=ad,dc=example,dc=com
ldap_group_object_class = group

ldap_access_order = expire
ldap_account_expire_policy = ad
ldap_force_upper_case_realm = true
ldap_referrals = false

krb5_realm = AD-REALM.EXAMPLE.COM
# required
krb5_canonicalize = false

```

7. Restart SSSD.

```
~]# service sssd restart
```

13.2.15. Domain Options: Setting Username Formats

One of the primary actions that SSSD performs is mapping a local system user to an identity in the remote identity provider. SSSD uses a combination of the user name and the domain back end name to create the login identity.

As long as they belong to different domains, SSSD can recognize different users with the same user name. For example, SSSD can successfully authenticate both **jsmith** in the **ldap.example.com** domain and **jsmith** in the **ldap.otherexample.com** domain.

The name format used to construct full user name is (optionally) defined universally in the **[sssd]** section of the configuration and can then be defined individually in each domain section.

Usernames for different services – LDAP, Samba, Active Directory, Identity Management, even the local system – all have different formats. The expression that SSSD uses to identify user name/domain name sets must be able to interpret names in different formats. This expression is set in the **re_expression** parameter.

In the global default, this filter constructs a name in the form *name@domain*:

```
(?P<name>[^@]+)@?(?P<domain>[^@]*$)
```



NOTE

The regular expression format is Python syntax.

The domain part may be supplied automatically, based on the domain name of the identity provider. Therefore, a user can log in as **jsmith** and if the user belongs to the LOCAL domain (for example), then his user name is interpreted by SSSD as **jsmith@LOCAL**.

However, other identity providers may have other formats. Samba, for example, has a very strict format so that user name must match the form `DOMAIN\username`. For Samba, then, the regular expression must be:

```
(?P<domain>[^\\]*?)\\?(?P<name>[^\\]+$)
```

Some providers, such as Active Directory, support multiple different name formats. Active Directory and Identity Management, for example, support three different formats by default:

- `username`
- `username@domain.name`
- `DOMAIN\username`

The default value for Active Directory and Identity Management providers, then, is a more complex filter that allows all three name formats:

```
(((?P<domain>[^\\]+)\\(?P<name>.+$))|((?P<name>[^@]+)@(?P<domain>.+$))|(^(?P<name>[^@\\]+)$))
```



NOTE

Requesting information with the fully-qualified name, such as `jsmith@ldap.example.com`, always returns the proper user account. If there are multiple users with the same user name in different domains, specifying only the user name returns the user for whichever domain comes first in the lookup order.

While **re_expression** is the most important method for setting user name formats, there are two other options which are useful for other applications.

Default Domain Name Value

The first sets a default domain name to be used with all users, **default_domain_suffix**. (This is a global setting, available in the **[sssd]** section only.) There may be a case where multiple domains are configured but only one stores user data and the others are used for host or service identities. Setting a default domain name allows users to log in with only their user name, not specifying the domain name (which would be required for users outside the primary domain).

```
[sssd]
...
default_domain_suffix = USERS.EXAMPLE.COM
```

Full Name Format for Output

The other parameter is related to **re_expression**, only instead of defining how to *interpret* a user name, it defines how to *print* an identified name. The **full_name_format** parameter sets how the user name and domain name (once determined) are displayed.

```
full_name_format = %1$s@%2$s
```

13.2.16. Domain Options: Enabling Offline Authentication

User identities are always cached, as well as information about the domain services. However, user

credentials are not cached by default. This means that SSSD always checks with the back end identity provider for authentication requests. If the identity provider is offline or unavailable, there is no way to process those authentication requests, so user authentication could fail.

It is possible to enable *offline credentials caching*, which stores credentials (after successful login) as part of the user account in the SSSD cache. Therefore, even if an identity provider is unavailable, users can still authenticate, using their stored credentials. Offline credentials caching is primarily configured in each individual domain entry, but there are some optional settings that can be set in the PAM service section, because credentials caching interacts with the local PAM service as well as the remote domain.

```
[domain/EXAMPLE]
cache_credentials = true
```

There are optional parameters that set when those credentials expire. Expiration is useful because it can prevent a user with a potentially outdated account or credentials from accessing local services indefinitely.

The credentials expiration itself is set in the PAM service, which processes authentication requests for the system.

```
[sssd]
services = nss,pam
...
[pam]
offline_credentials_expiration = 3
...
[domain/EXAMPLE]
cache_credentials = true
...
```

offline_credentials_expiration sets the number of days after a successful login that a single credentials entry for a user is preserved in cache. Setting this to zero (0) means that entries are kept forever.

While not related to the credentials cache specifically, each domain has configuration options on when individual user and service caches expire:

- **account_cache_expiration** sets the number of days after a successful login that the entire user account entry is removed from the SSSD cache. This must be equal to or longer than the individual offline credentials cache expiration period.
- **entry_cache_timeout** sets a validity period, in seconds, for all entries stored in the cache before SSSD requests updated information from the identity provider. There are also individual cache timeout parameters for group, service, netgroup, sudo, and autofs entries; these are listed in the **sssd.conf** man page. The default time is 5400 seconds (90 minutes).

For example:

```
[sssd]
services = nss,pam
...
[pam]
offline_credentials_expiration = 3
```

```

...
[domain/EXAMPLE]
cache_credentials = true
account_cache_expiration = 7
entry_cache_timeout = 14400
...

```

13.2.17. Domain Options: Setting Password Expirations

Password policies generally set an expiration time, after which passwords expire and must be replaced. Password expiration policies are evaluated on the server side through the identity provider, then a warning can be processed and displayed in SSSD through its PAM service.

There are two ways to display password expiration warnings:

- The ***pam_pwd_expiration_warning*** parameter defines the global default setting for all domains on how far in advance of the password expiration to display a warning. This is set for the PAM service.
- The ***pwd_expiration_warning*** parameter defines the per-domain setting on how far in advance of the password expiration to display a warning.

When using a domain-level password expiration warning, an authentication provider (***auth_provider***) must also be configured for the domain.

For example:

```

[sssd]
services = nss,pam
...
[pam]
pam_pwd_expiration_warning = 3
...
[domain/EXAMPLE]
id_provider = ipa
auth_provider = ipa
pwd_expiration_warning = 7

```

The password expiration warning must be sent from the server to SSSD for the warning to be displayed. If no password warning is sent from the server, no message is displayed through SSSD, even if the password expiration time is within the period set in SSSD.

If the password expiration warning is not set in SSSD or is set to **0**, then the SSSD password warning filter is not applied and the server-side password warning is automatically displayed.



NOTE

As long as the password warning is sent from the server, the PAM or domain password expirations in effect override the password warning settings on the back end identity provider. For example, consider a back end identity provider that has the warning period set at 28 days, but the PAM service in SSSD has it set to 7 days. The provider sends the warning to SSSD starting at 28 days, but the warning is not displayed locally until 7 days, according to the password expiration set in the SSSD configuration.

Password Expiration Warnings for Non-Password Authentication

By default, password expiration is verified only if the user enters the password during authentication. However, you can configure SSSD to perform the expiration check and display the warning even when a non-password authentication method is used, for example, during SSH login.

To enable password expiration warnings with non-password authentication methods:

1. Make sure the **access_provider** parameter is set to **ldap** in the **sssd.conf** file.
2. Make sure the **ldap_pwd_policy** parameter is set in **sssd.conf**. In most situations, the appropriate value is **shadow**.
3. Add one of the following **pwd_expire_*** values to the **ldap_access_order** parameter in **sssd.conf**. If the password is about to expire, each one of these values only displays the expiration warning. In addition:
 - o **pwd_expire_policy_reject** prevents the user from logging in if the password is already expired.
 - o **pwd_expire_policy_warn** allows the user to log in even if the password is already expired.
 - o **pwd_expire_policy_renew** prompts the user to immediately change the password if the user attempts to log in with an expired password.

For example:

```
[domain/EXAMPLE]
access_provider = ldap
ldap_pwd_policy = shadow
ldap_access_order = pwd_expire_policy_warn
```

For more details on using **ldap_access_order** and its values, see the `sssd-ldap(5)` man page.

13.2.18. Domain Options: Using DNS Service Discovery

DNS service discovery, defined in [RFC 2782](#), allows applications to check the SRV records in a given domain for certain services of a certain type; it then returns any servers discovered of that type.

With SSSD, the identity and authentication providers can either be explicitly defined (by IP address or host name) or they can be discovered dynamically, using service discovery. If no provider server is listed – for example, if **id_provider = ldap** is set without a corresponding **ldap_uri** parameter – then discovery is automatically used.

The DNS discovery query has this format:

```
_service._protocol.domain
```

For example, a scan for an LDAP server using TCP in the **example.com** domain looks like this:

```
_ldap._tcp.example.com
```



NOTE

For every service with which to use service discovery, add a special DNS record to the DNS server:

```
_service._protocol._domain TTL priority weight port hostname
```

For SSSD, the service type is LDAP by default, and almost all services use TCP (except for Kerberos, which starts with UDP). For service discovery to be enabled, the only thing that is required is the domain name. The default is to use the domain portion of the machine host name, but another domain can be specified (using the **dns_discovery_domain** parameter).

So, by default, no additional configuration needs to be made for service discovery – with one exception. The password change provider has server discovery disabled by default, and it must be explicitly enabled by setting a service type.

```
[domain/EXAMPLE]
...
chpass_provider = ldap
ldap_chpass_dns_service_name = ldap
```

While no configuration is necessary, it is possible for server discovery to be customized by using a different DNS domain (**dns_discovery_domain**) or by setting a different service type to scan for. For example:

```
[domain/EXAMPLE]
id_provider = ldap

dns_discovery_domain = corp.example.com
ldap_dns_service_name = ldap

chpass_provider = krb5
ldap_chpass_dns_service_name = kerberos
```

Lastly, service discovery is never used with backup servers; it is only used for the primary server for a provider. What this means is that discovery can be used initially to locate a server, and then SSSD can fall back to using a backup server. To use discovery for the primary server, use **_srv_** as the primary server value, and then list the backup servers. For example:

```
[domain/EXAMPLE]
id_provider = ldap
ldap_uri = _srv_
ldap_backup_uri = ldap://ldap2.example.com

auth_provider = krb5
krb5_server = _srv_
krb5_backup_server = kdc2.example.com

chpass_provider = krb5
```

```
ldap_chpass_dns_service_name = kerberos
ldap_chpass_uri = _srv_
ldap_chpass_backup_uri = kdc2.example.com
```

**NOTE**

Service discovery cannot be used with backup servers, only primary servers.

If a DNS lookup fails to return an IPv4 address for a host name, SSSD attempts to look up an IPv6 address before returning a failure. This only ensures that the asynchronous resolver identifies the correct address.

The host name resolution behavior is configured in the ***lookup family order*** option in the ***sssd.conf*** configuration file.

13.2.19. Domain Options: Using IP Addresses in Certificate Subject Names (LDAP Only)

Using an IP address in the ***ldap_uri*** option instead of the server name may cause the TLS/SSL connection to fail. TLS/SSL certificates contain the server name, not the IP address. However, the *subject alternative name* field in the certificate can be used to include the IP address of the server, which allows a successful secure connection using an IP address.

Procedure 13.8. Using IP Addresses in Certificate Subject Names

- Convert an existing certificate into a certificate request. The signing key (***-signkey***) is the key of the issuer of whatever CA originally issued the certificate. If this is done by an external CA, it requires a separate PEM file; if the certificate is self-signed, then this is the certificate itself. For example:

```
openssl x509 -x509toreq -in old_cert.pem -out req.pem -signkey key.pem
```

With a self-signed certificate:

```
openssl x509 -x509toreq -in old_cert.pem -out req.pem -signkey old_cert.pem
```

- Edit the **/etc/pki/tls/openssl.cnf** configuration file to include the server's IP address under the **[v3_ca]** section:

```
subjectAltName = IP:10.0.0.10
```

- Use the generated certificate request to generate a new self-signed certificate with the specified IP address:

```
openssl x509 -req -in req.pem -out new_cert.pem -extfile ./openssl.cnf -extensions v3_ca -signkey old_cert.pem
```

The ***-extensions*** option sets which extensions to use with the certificate. For this, it should be *v3_ca* to load the appropriate section.

- Copy the private key block from the **old_cert.pem** file into the **new_cert.pem** file to keep all relevant information in one file.

When creating a certificate through the **certutil** utility provided by the `nss-tools` package, note that **certutil** supports DNS subject alternative names for certificate creation only.

13.2.20. Creating Domains: Proxy

A proxy with SSSD is just a relay, an intermediary configuration. SSSD connects to its proxy service, and then that proxy loads the specified libraries. This allows SSSD to use some resources that it otherwise would not be able to use. For example, SSSD only supports LDAP and Kerberos as authentication providers, but using a proxy allows SSSD to use alternative authentication methods like a fingerprint scanner or smart card.

Table 13.9. Proxy Domain Configuration Parameters

Parameter	Description
proxy_pam_target	<p>Specifies the target to which PAM must proxy as an authentication provider. The PAM target is a file containing PAM stack information in the default PAM directory, /etc/pam.d/. This is used to proxy an authentication provider.</p> <p> IMPORTANT</p> <p>Ensure that the proxy PAM stack does <i>not</i> recursively include pam_sss.so.</p>
proxy_lib_name	Specifies which existing NSS library to proxy identity requests through. This is used to proxy an identity provider.

Example 13.10. Proxy Identity and Kerberos Authentication

The proxy library is loaded using the **proxy_lib_name** parameter. This library can be anything as long as it is compatible with the given authentication service. For a Kerberos authentication provider, it must be a Kerberos-compatible library, like NIS.

```
[domain/PROXY_KRB5]
auth_provider = krb5
krb5_server = kdc.example.com
krb5_realm = EXAMPLE.COM

id_provider = proxy
proxy_lib_name = nis
cache_credentials = true
```

Example 13.11. LDAP Identity and Proxy Authentication

The proxy library is loaded using the **proxy_pam_target** parameter. This library must be a PAM module that is compatible with the given identity provider. For example, this uses a PAM fingerprint module with LDAP:

```
[domain/LDAP_PROXY]
id_provider = ldap
```

```

ldap_uri = ldap://example.com
ldap_search_base = dc=example,dc=com

auth_provider = proxy
proxy_pam_target = sssdpamproxy
cache_credentials = true

```

After the SSSD domain is configured, make sure that the specified PAM files are configured. In this example, the target is **sssdpamproxy**, so create a **/etc/pam.d/sssdpamproxy** file and load the PAM/LDAP modules:

```

auth      required    pam_frprint.so
account   required    pam_frprint.so
password  required    pam_frprint.so
session   required    pam_frprint.so

```

Example 13.12. Proxy Identity and Authentication

SSSD can have a domain with both identity and authentication proxies. The only configuration given then are the proxy settings, **proxy_pam_target** for the authentication PAM module and **proxy_lib_name** for the service, like NIS or LDAP.

This example illustrates a possible configuration, but this is not a realistic configuration. If LDAP is used for identity and authentication, then both the identity and authentication providers should be set to the LDAP configuration, not a proxy.

```

[domain/PROXY_PROXY]
auth_provider = proxy
id_provider = proxy
proxy_lib_name = ldap
proxy_pam_target = sssdproxyldap
cache_credentials = true

```

Once the SSSD domain is added, then update the system settings to configure the proxy service:

1. Create a **/etc/pam.d/sssdproxyldap** file which requires the **pam_ldap.so** module:

```

auth      required    pam_ldap.so
account   required    pam_ldap.so
password  required    pam_ldap.so
session   required    pam_ldap.so

```

2. Make sure the nss-pam-ldapd package is installed.

```

~]# yum install nss-pam-ldapd

```

3. Edit the **/etc/nslcd.conf** file, the configuration file for the LDAP name service daemon, to contain the information for the LDAP directory:

```

uid nslcd
gid ldap
uri ldaps://ldap.example.com:636

```

```
base dc=example,dc=com
ssl on
tls_cacertdir /etc/openldap/cacerts
```

13.2.21. Creating Domains: Kerberos Authentication

Both LDAP and proxy identity providers can use a separate Kerberos domain to supply authentication. Configuring a Kerberos authentication provider requires the *key distribution center* (KDC) and the Kerberos domain. All of the principal names must be available in the specified identity provider; if they are not, SSSD constructs the principals using the format *username@REALM*.



NOTE

Kerberos can only provide authentication; it cannot provide an identity database.

SSSD assumes that the Kerberos KDC is also a Kerberos kadmin server. However, production environments commonly have multiple, read-only replicas of the KDC and only a single kadmin server. Use the **krb5_kpasswd** option to specify where the password changing service is running or if it is running on a non-default port. If the **krb5_kpasswd** option is not defined, SSSD tries to use the Kerberos KDC to change the password.

The basic Kerberos configuration options are listed in [Table 13.10, “Kerberos Authentication Configuration Parameters”](#). The **sssd-krb5(5)** man page has more information about Kerberos configuration options.

Example 13.13. Basic Kerberos Authentication

```
# A domain with identities provided by LDAP and authentication by Kerberos
[domain/KRBDOMAIN]
id_provider = ldap
chpass_provider = krb5
ldap_uri = ldap://ldap.example.com
ldap_search_base = dc=example,dc=com
ldap-tls_reqcert = demand
ldap_tls_cacert = /etc/pki/tls/certs/ca-bundle.crt

auth_provider = krb5
krb5_server = kdc.example.com
krb5_backup_server = kerberos.example.com
krb5_realm = EXAMPLE.COM
krb5_kpasswd = kerberos.admin.example.com
krb5_auth_timeout = 15
```

Example 13.14. Setting Kerberos Ticket Renewal Options

The Kerberos authentication provider, among other tasks, requests ticket granting tickets (TGT) for users and services. These tickets are used to generate other tickets dynamically for specific services, as accessed by the ticket principal (the user).

The TGT initially granted to the user principal is valid only for the lifetime of the ticket (by default, whatever is configured in the configured KDC). After that, the ticket cannot be renewed or extended.

However, not renewing tickets can cause problems with some services when they try to access a service in the middle of operations and their ticket has expired.

Kerberos tickets are not renewable by default, but ticket renewal can be enabled using the **krb5_renewable_lifetime** and **krb5_renew_interval** parameters.

The lifetime for a ticket is set in SSSD with the **krb5_lifetime** parameter. This specifies how long a single ticket is valid, and overrides any values in the KDC.

Ticket renewal itself is enabled in the **krb5_renewable_lifetime** parameter, which sets the maximum lifetime of the ticket, counting all renewals.

For example, the ticket lifetime is set at one hour and the renewable lifetime is set at 24 hours:

```
krb5_lifetime = 1h
krb5_renewable_lifetime = 1d
```

This means that the ticket expires every hour and can be renewed continually up to one day.

The lifetime and renewable lifetime values can be in seconds (s), minutes (m), hours (h), or days (d).

The other option – which must also be set for ticket renewal – is the **krb5_renew_interval** parameter, which sets how frequently SSSD checks to see if the ticket needs to be renewed. At half of the ticket lifetime (whatever that setting is), the ticket is renewed automatically. (This value is always in seconds.)

```
krb5_lifetime = 1h
krb5_renewable_lifetime = 1d
krb5_renew_interval = 60s
```



NOTE

If the **krb5_renewable_lifetime** value is not set or the **krb5_renew_interval** parameter is not set or is set to zero (0), then ticket renewal is disabled. Both **krb5_renewable_lifetime** and **krb5_renew_interval** are required for ticket renewal to be enabled.

Table 13.10. Kerberos Authentication Configuration Parameters

Parameter	Description
chpass_provider	Specifies which service to use for password change operations. This is assumed to be the same as the authentication provider. To use Kerberos, set this to <i>krb5</i> .
krb5_server	Gives the primary Kerberos server, by IP address or host names, to which SSSD will connect.

Parameter	Description
krb5_backup_server	<p>Gives a comma-separated list of IP addresses or host names of Kerberos servers to which SSSD will connect if the primary server is not available. The list is given in order of preference, so the first server in the list is tried first.</p> <p>After an hour, SSSD will attempt to reconnect to the primary service specified in the krb5_server parameter.</p> <p>When using service discovery for KDC or kpasswd servers, SSSD first searches for DNS entries that specify UDP as the connection protocol, and then falls back to TCP.</p>
krb5_realm	Identifies the Kerberos realm served by the KDC.
krb5_lifetime	Requests a Kerberos ticket with the specified lifetime in seconds (s), minutes (m), hours (h) or days (d).
krb5_renewable_lifetime	Requests a renewable Kerberos ticket with a total lifetime that is specified in seconds (s), minutes (m), hours (h) or days (d).
krb5_renew_interval	Sets the time, in seconds, for SSSD to check if tickets should be renewed. Tickets are renewed automatically once they exceed half their lifetime. If this option is missing or set to zero, then automatic ticket renewal is disabled.
krb5_store_password_if_offline	Sets whether to store user passwords if the Kerberos authentication provider is offline, and then to use that cache to request tickets when the provider is back online. The default is false , which does not store passwords.
krb5_kpasswd	Lists alternate Kerberos kadmin servers to use if the change password service is not running on the KDC.

Parameter	Description
krb5_ccname_template	<p>Gives the directory to use to store the user's credential cache. This can be templated, and the following tokens are supported:</p> <ul style="list-style-type: none"> ● $\%u$, the user's login name ● $\%U$, the user's login UID ● $\%p$, the user's principal name ● $\%r$, the realm name ● $\%h$, the user's home directory ● $\%d$, the value of the krb5ccache_dir parameter ● $\%P$, the process ID of the SSSD client. ● $\%\%$, a literal percent sign (%) ● XXXXXX, a string at the end of the template which instructs SSSD to create a unique filename safely <p>For example:</p> <pre>krb5_ccname_template = FILE:%d/krb5cc_%U_XXXXXX</pre>
krb5_ccachedir	Specifies the directory to store credential caches. This can be templated, using the same tokens as krb5_ccname_template , except for %d and %P . If %u , %U , %p , or %h are used, then SSSD creates a private directory for each user; otherwise, it creates a public directory.
krb5_auth_timeout	Gives the time, in seconds, before an online authentication or change password request is aborted. If possible, the authentication request is continued offline. The default is 15 seconds.

13.2.22. Creating Domains: Access Control

SSSD provides a rudimentary access control for domain configuration, allowing either simple user allow/deny lists or using the LDAP back end itself.

Using the Simple Access Provider

The *Simple Access Provider* allows or denies access based on a list of user names or groups.

The Simple Access Provider is a way to restrict access to certain, specific machines. For example, if a company uses laptops, the Simple Access Provider can be used to restrict access to only a specific user or a specific group, even if a different user authenticated successfully against the same authentication provider.

The most common options are **simple_allow_users** and **simple_allow_groups**, which grant access explicitly to specific users (either the given users or group members) and deny access to everyone else. It is also possible to create deny lists (which deny access only to explicit people and implicitly allow everyone else access).

The Simple Access Provider adheres to the following four rules to determine which users should or should not be granted access:

- If both the allow and deny lists are empty, access is granted.
- If any list is provided, allow rules are evaluated first, and then deny rules. Practically, this means that deny rules supersede allow rules.
- If an allowed list is provided, then all users are denied access unless they are in the list.
- If only deny lists are provided, then all users are allowed access unless they are in the list.

This example grants access to two users and anyone who belongs to the IT group; implicitly, all other users are denied:

```
[domain/example.com]
access_provider = simple
simple_allow_users = jsmith,bjensen
simple_allow_groups = itgroup
```



NOTE

The LOCAL domain in SSSD does not support **simple** as an access provider.

Other options are listed in the **sssd-simple** man page, but these are rarely used.

Using the Access Filters

An LDAP, Active Directory, or Identity Management server can provide access control rules for a domain. The associated options (**ldap_access_filter** for LDAP and IdM and **ad_access_filter** for AD) specify which users are granted access to the specified host. The user filter must be used or all users are denied access. See the examples below:

```
[domain/example.com]
access_provider = ldap
ldap_access_filter = memberOf=cn=allowedusers,ou=Groups,dc=example,dc=com
```

```
[domain/example.com]
access_provider = ad
ad_access_filter = memberOf=cn=allowedusers,ou=Groups,dc=example,dc=com
```



NOTE

Offline caching for LDAP access providers is limited to determining whether the user's last online login attempt was successful. Users that were granted access during their last login will continue to be granted access while offline.

SSSD can also check results by the **authorizedService** or **host** attribute in an entry. In fact, all options – LDAP filter, **authorizedService**, and **host** – can be evaluated, depending on the user entry and the configuration. The **ldap_access_order** parameter lists all access control methods to use, in order of how they should be evaluated.

```
[domain/example.com]
access_provider = ldap
```

```
ldap_access_filter = memberOf=cn=allowedusers,ou=Groups,dc=example,dc=com
ldap_access_order = filter, host, authorized_service
```

The attributes in the user entry to use to evaluate authorized services or allowed hosts can be customized. Additional access control parameters are listed in the **sssd-ldap(5)** man page.

13.2.23. Creating Domains: Primary Server and Backup Servers

Identity and authentication providers for a domain can be configured for automatic failover. SSSD attempts to connect to the specified, primary server first. If that server cannot be reached, then SSSD then goes through the listed backup servers, in order.



NOTE

SSSD tries to connect to the primary server every 30 seconds, until the connection can be re-established, and then switches from the backup to the primary.

All of the major service areas have optional settings for primary and backup servers^[3].

Table 13.11. Primary and Secondary Server Parameters

Service Area	Primary Server Attribute	Backup Server Attribute
LDAP identity provider	ldap_uri	ldap_backup_uri
Active Directory identity provider	ad_server	ad_backup_server
Identity Management (IdM or IPA) identity provider	ipa_server	ipa_backup_server
Kerberos authentication provider	krb5_server	krb5_backup_server
Kerberos authentication provider	krb5_server	krb5_backup_server
Password change provider	ldap_chpass_uri	ldap_chpass_backup_uri

One and only one server can be set as the primary server. (And, optionally, the primary server can be set to service discovery, using **_srv_** rather than a host name.) Multiple backup servers can be set, in a comma-separated list. The backup server list is in order of preference, so the first server listed is tried first.

```
[domain/EXAMPLE]
id_provider = ad
ad_server = ad.example.com
ad_backup_server = ad1.example.com, ad-backup.example.com
```

13.2.24. Installing SSSD Utilities

Additional tools to handle the SSSD cache, user entries, and group entries are contained in the **sssd-tools** package. This package is not required, but it is useful to install to help administer user accounts.

```
~]# yum install sssd-tools
```



NOTE

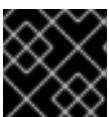
The `sssd-tools` package is provided by the Optional subscription channel. See [Section 8.4.8, "Adding the Optional and Supplementary Repositories"](#) for more information on Red Hat additional channels.

13.2.25. SSSD and UID and GID Numbers

When a user is created – using system tools such as `useradd` or through an application such as Red Hat Identity Management or other client tools – the user is automatically assigned a user ID number and a group ID number.

When the user logs into a system or service, SSSD caches that user name with the associated UID/GID numbers. The UID number is then used as the identifying key for the user. If a user with the same name but a different UID attempts to log into the system, then SSSD treats it as two different users with a name collision.

What this means is that SSSD does not recognize UID number changes. It interprets it as a different and new user, not an existing user with a different UID number. If an existing user changes the UID number, that user is prevented from logging into SSSD and associated services and domains. This also has an impact on any client applications which use SSSD for identity information; the user with the conflict will not be found or accessible to those applications.



IMPORTANT

UID/GID changes are not supported in SSSD.

If a user for some reason has a changed UID/GID number, then the SSSD cache must be cleared for that user before that user can log in again. For example:

```
~]# sss_cache -u jsmith
```

Cleaning the SSSD cache is covered in [the section called “Purging the SSSD Cache”](#).

13.2.26. Creating Local System Users

There can be times when it is useful to seed users into the SSSD database rather than waiting for users to login and be added.



NOTE

Adding user accounts manually requires the `sssd-tools` package to be installed.

When creating new system users, it is possible to create a user within the SSSD local identity provider domain. This can be useful for creating new system users, for troubleshooting SSSD configuration, or for creating specialized or nested groups.

New users can be added using the `sss_useradd` command.

At its most basic, the `sss_useradd` command only requires the new user name.

```
~]# sss_useradd jsmith
```

There are other options (listed in the **sss_useradd(8)** man page) which can be used to set attributes on the account, like the UID and GID, the home directory, or groups which the user belongs to.

```
~]# sss_useradd --UID 501 --home /home/jsmith --groups admin,dev-group jsmith
```

13.2.27. Seeding Users into the SSSD Cache During Kickstart



NOTE

Adding user accounts manually requires the `sssd-tools` package to be installed.

With SSSD, users in a remote domain are not available in a local system until that identity is retrieved from the identity provider. However, some network interfaces are not available until a user has logged in – which is not possible if the user identity is somewhere over the network. In that case, it is possible to seed the SSSD cache with that user identity, associated with the appropriate domain, so that the user can log in locally and activate the appropriate interfaces.

This is done using the **sss_seed** utility:

```
ssssd_seed --domain EXAMPLE.COM --username testuser --password-file /tmp/sssd-pwd.txt
```

This utility requires options that identify, at a minimum, the user name, domain name, and password.

- **--domain** gives the domain name from the SSSD configuration. This domain must already exist in the SSSD configuration.
- **--username** for the short name of the user account.
- **--password-file** for the path and name of a file containing a temporary password for the seed entry. If the user account already exists in the SSSD cache, then the temporary password in this file overwrites the stored password in the SSSD cache.

Additional account configuration options are listed in the **sss_seed(8)** man page.

This would almost always be run as part of a kickstart or automated setup, so it would be part of a larger set of scripts, which would also enable SSSD, set up an SSSD domain, and create the password file. For example:

```
function make_ssdd {
cat <<- _EOF_
[sssd]
domains = LOCAL
services = nss,pam

[nss]

[pam]

[domain/LOCAL]
id_provider = local
auth_provider = local
```

```

access_provider = permit

_EOF_
}

make_sssd >> /etc/sssd/sssd.conf

authconfig --enablesssd --enablesssdauth --update

function make_pwdfile {
cat <<1 _EOF_
password
_EOF_
}

make_pwdfile >> /tmp/sssd-pwd.txt

sss_seed --domain EXAMPLE.COM --username testuser --password-file /tmp/sssd-pwd.txt

```

13.2.28. Managing the SSSD Cache

SSSD can define multiple domains of the same type and different types of domain. SSSD maintains a separate database file for each domain, meaning each domain has its own cache. These cache files are stored in the **/var/lib/sss/db** directory.

Purging the SSSD Cache

As LDAP updates are made to the identity provider for the domains, it can be necessary to clear the cache to reload the new information quickly.

The cache purge utility, **sss_cache**, invalidates records in the SSSD cache for a user, a domain, or a group. Invalidating the current records forces the cache to retrieve the updated records from the identity provider, so changes can be realized quickly.



NOTE

This utility is included with SSSD in the `sssd` package.

Most commonly, this is used to clear the cache and update all records:

```
~]# sss_cache -E
```

The **sss_cache** command can also clear all cached entries for a particular domain:

```
~]# sss_cache -Ed LDAP1
```

If the administrator knows that a specific record (user, group, or netgroup) has been updated, then **sss_cache** can purge the records for that specific account and leave the rest of the cache intact:

```
~]# sss_cache -u jsmith
```

Table 13.12. Common sss_cache Options

Short Argument	Long Argument	Description
-E	--everything	Invalidates all cached entries with the exception of sudo rules.
-d <i>name</i>	--domain <i>name</i>	Invalidates cache entries for users, groups, and other entries only within the specified domain.
-G	--groups	Invalidates all group records. If -g is also used, -G takes precedence and -g is ignored.
-g <i>name</i>	--group <i>name</i>	Invalidates the cache entry for the specified group.
-N	--netgroups	Invalidates cache entries for all netgroup cache records. If -n is also used, -N takes precedence and -n is ignored.
-n <i>name</i>	--netgroup <i>name</i>	Invalidates the cache entry for the specified netgroup.
-U	--users	Invalidates cache entries for all user records. If the -u option is also used, -U takes precedence and -u is ignored.
-u <i>name</i>	--user <i>name</i>	Invalidates the cache entry for the specified user.

Deleting Domain Cache Files

All cache files are named for the domain. For example, for a domain named **exampleldap**, the cache file is named **cache_exampleldap ldb**.

Be careful when you delete a cache file. This operation has significant effects:

- Deleting the cache file deletes all user data, both identification and cached credentials. Consequently, do not delete a cache file unless the system is online and can authenticate with a user name against the domain's servers. Without a credentials cache, offline authentication will fail.
- If the configuration is changed to reference a different identity provider, SSSD will recognize users from both providers until the cached entries from the original provider time out.

It is possible to avoid this by purging the cache, but the better option is to use a different domain name for the new provider. When SSSD is restarted, it creates a new cache file with the new name and the old file is ignored.

13.2.29. Downgrading SSSD

When downgrading – either downgrading the version of SSSD or downgrading the operating system itself – then the existing SSSD cache needs to be removed. If the cache is not removed, then SSSD process is dead but a PID file remains. The SSSD logs show that it cannot connect to any of its associated domains because the cache version is unrecognized.

(Wed Nov 28 21:25:50 2012) [sssd] [sysdb_domain_init_internal] (0x0010): Unknown DB version [0.14], expected [0.10] for domain AD!

Users are then no longer recognized and are unable to authenticate to domain services and hosts.

After downgrading the SSSD version:

1. Delete the existing cache database files.

```
~]# rm -rf /var/lib/sssd/db/*
```

2. Restart the SSSD process.

```
~]# service sssd restart
Stopping sssd: [FAILED]
Starting sssd: [ OK ]
```

13.2.30. Using NSCD with SSSD

SSSD is not designed to be used with the NSCD daemon. Even though SSSD does not directly conflict with NSCD, using both services can result in unexpected behavior, especially with how long entries are cached.

The most common evidence of a problem is conflicts with NFS. When using Network Manager to manage network connections, it may take several minutes for the network interface to come up. During this time, various services attempt to start. If these services start before the network is up and the DNS servers are available, these services fail to identify the forward or reverse DNS entries they need. These services will read an incorrect or possibly empty **resolv.conf** file. This file is typically only read once, and so any changes made to this file are not automatically applied. This can cause NFS locking to fail on the machine where the NSCD service is running, unless that service is manually restarted.

To avoid this problem, enable caching for hosts and services in the **/etc/nscd.conf** file and rely on the SSSD cache for the **passwd**, **group**, and **netgroup** entries.

Change the **/etc/nscd.conf** file:

```
enable-cache hosts yes
enable-cache passwd no
enable-cache group no
enable-cache netgroup no
```

With NSCD answering hosts requests, these entries will be cached by NSCD and returned by NSCD during the boot process. All other entries are handled by SSSD.

13.2.31. Troubleshooting SSSD

- the section called “Setting Debug Logs for SSSD Domains”
- the section called “Checking SSSD Log Files”
- the section called “Problems with SSSD Configuration”

Setting Debug Logs for SSSD Domains

Each domain sets its own debug log level. Increasing the log level can provide more information about problems with SSSD or with the domain configuration.

To change the log level, set the **debug_level** parameter for each section in the **sssd.conf** file for which to produce extra logs. For example:

```
[domain/LDAP]
cache_credentials = true
debug_level = 9
```

Table 13.13. Debug Log Levels

Level	Description
0	Fatal failures. Anything that would prevent SSSD from starting up or causes it to cease running.
1	Critical failures. An error that doesn't kill the SSSD, but one that indicates that at least one major feature is not going to work properly.
2	Serious failures. An error announcing that a particular request or operation has failed.
3	Minor failures. These are the errors that would percolate down to cause the operation failure of 2.
4	Configuration settings.
5	Function data.
6	Trace messages for operation functions.
7	Trace messages for internal control functions.
8	Contents of function-internal variables that may be interesting.
9	Extremely low-level tracing information.



NOTE

In versions of SSSD older than 1.8, debug log levels could be set globally in the **[sssd]** section. Now, each domain and service must configure its own debug log level.

To copy the global SSSD debug log levels into each configuration area in the SSSD configuration file, use the **sssd_update_debug_levels.py** script.

```
python -m SSSDConfig.sssd_update_debug_levels.py
```

Checking SSSD Log Files

SSSD uses a number of log files to report information about its operation, located in the **/var/log/sssd/** directory. SSSD produces a log file for each domain, as well as an **sssd_pam.log** and an **sssd_nss.log** file.

Additionally, the **/var/log/secure** file logs authentication failures and the reason for the failure.

Problems with SSSD Configuration

Q: SSSD fails to start

- A: SSSD requires that the configuration file be properly set up, with all the required entries, before the daemon will start.

SSSD requires at least one properly configured domain before the service will start. Without a domain, attempting to start SSSD returns an error that no domains are configured:

```
# sssd -d4

[sssd] [ldb] (3): server_sort:Unable to register control with rootdse!
[sssd] [confdb_get_domains] (0): No domains configured, fatal error!
[sssd] [get_monitor_config] (0): No domains configured.
```

Edit the **/etc/sssd/sssd.conf** file and create at least one domain.

SSSD also requires at least one available service provider before it will start. If the problem is with the service provider configuration, the error message indicates that there are no services configured:

```
[sssd] [get_monitor_config] (0): No services configured!
```

Edit the **/etc/sssd/sssd.conf** file and configure at least one service provider.



IMPORTANT

SSSD requires that service providers be configured as a comma-separated list in a single **services** entry in the **/etc/sssd/sssd.conf** file. If services are listed in multiple entries, only the last entry is recognized by SSSD.

- Q: I don't see any groups with 'id' or group members with 'getent group'.

- A: This may be due to an incorrect **ldap_schema** setting in the **[domain/DOMAINNAME]** section of **sssd.conf**.

SSSD supports RFC 2307 and RFC 2307bis schema types. By default, SSSD uses the more common RFC 2307 schema.

The difference between RFC 2307 and RFC 2307bis is the way which group membership is stored in the LDAP server. In an RFC 2307 server, group members are stored as the multi-valued **memberuid** attribute, which contains the name of the users that are members. In an RFC2307bis server, group members are stored as the multi-valued **member** or **uniqueMember** attribute which contains the DN of the user or group that is a member of this group. RFC2307bis allows nested groups to be maintained as well.

If group lookups are not returning any information:

1. Set **ldap_schema** to **rfc2307bis**.
2. Delete **/var/lib/sss/db/cache_DOMAINNAME ldb**.
3. Restarting SSSD.

If that doesn't work, add this line to **sssd.conf**:

```
    | ldap_group_name = uniqueMember
```

Then delete the cache and restart SSSD again.

Q: Authentication fails against LDAP.

- A:** To perform authentication, SSSD requires that the communication channel be encrypted. This means that if **sssd.conf** is configured to connect over a standard protocol (**ldap://**), it attempts to encrypt the communication channel with Start TLS. If **sssd.conf** is configured to connect over a secure protocol (**ldaps://**), then SSSD uses SSL.

This means that the LDAP server must be configured to run in SSL or TLS. TLS must be enabled for the standard LDAP port (389) or SSL enabled on the secure LDAPS port (636). With either SSL or TLS, the LDAP server must also be configured with a valid certificate trust.

An invalid certificate trust is one of the most common issues with authenticating against LDAP. If the client does not have proper trust of the LDAP server certificate, it is unable to validate the connection, and SSSD refuses to send the password. The LDAP protocol requires that the password be sent in plaintext to the LDAP server. Sending the password in plaintext over an unencrypted connection is a security problem.

If the certificate is not trusted, a **syslog** message is written, indicating that TLS encryption could not be started. The certificate configuration can be tested by checking if the LDAP server is accessible apart from SSSD. For example, this tests an anonymous bind over a TLS connection to **test.example.com**:

```
    | $ ldapsearch -x -ZZ -h test.example.com -b dc=example,dc=com
```

If the certificate trust is not properly configured, the test fails with this error:

```
    | ldap_start_tls: Connect error (-11) additional info: TLS error -8179:Unknown code __f 13
```

To trust the certificate:

1. Obtain a copy of the public CA certificate for the certificate authority used to sign the LDAP server certificate and save it to the local system.
2. Add a line to the **sssd.conf** file that points to the CA certificate on the filesystem.

```
    |     | ldap_tls_cacert = /path/to/cacert
```

3. If the LDAP server uses a self-signed certificate, remove the **ldap_tls_reqcert** line from the **sssd.conf** file.

This parameter directs SSSD to trust any certificate issued by the CA certificate, which is a security risk with a self-signed CA certificate.

Q: Connecting to LDAP servers on non-standard ports fail.

- A:** When running SELinux in enforcing mode, the client's SELinux policy has to be modified to connect to the LDAP server over the non-standard port. For example:

```
    | # semanage port -a -t ldap_port_t -p tcp 1389
```

Q: NSS fails to return user information

A: This usually means that SSSD cannot connect to the NSS service.

Ensure that NSS is running:

```
# service sssd status
```

If NSS is running, make sure that the provider is properly configured in the [**nss**] section of the **/etc/sssd/sssd.conf** file. Especially check the **filter_users** and **filter_groups** attributes.

Make sure that NSS is included in the list of services that SSSD uses.

Check the configuration in the **/etc/nsswitch.conf** file.

Q: NSS returns incorrect user information

A: If searches are returning the incorrect user information, check that there are not conflicting user names in separate domains. When there are multiple domains, set the **use_fully_qualified_domains** attribute to **true** in the **/etc/sssd/sssd.conf** file. This differentiates between different users in different domains with the same name.

Q: Setting the password for the local SSSD user prompts twice for the password

A: When attempting to change a local SSSD user's password, it may prompt for the password twice:

```
[root@clientF11 tmp]# passwd user1000
Changing password for user user1000.
New password:
Retype new password:
New Password:
Reenter new Password:
passwd: all authentication tokens updated successfully.
```

This is the result of an incorrect PAM configuration. Ensure that the **use_authok** option is correctly configured in your **/etc/pam.d/system-auth** file.

Q: I am trying to use sudo rules with an Identity Management (IPA) provider, but no sudo rules are being found, even though sudo is properly configured.

A: The SSSD client can successfully authenticate to the Identity Management server, and it is properly searching the LDAP directory for sudo rules. However, it is showing that no rules exist. For example, in the logs:

```
(Thu Jun 21 10:37:47 2012) [sssd[be[ipa.test]]] [sdap_sudo_load_sudoers_process] (0x0400):
Receiving sudo rules with base [ou=sudoers,dc=ipa,dc=test]
(Thu Jun 21 10:37:47 2012) [sssd[be[ipa.test]]] [sdap_sudo_load_sudoers_done] (0x0400):
Received 0 rules
(Thu Jun 21 10:37:47 2012) [sssd[be[ipa.test]]] [sdap_sudo_purge_sudoers] (0x0400): Purging
SUDOers cache of user's [admin] rules
```

```
(Thu Jun 21 10:37:47 2012) [sssd[be[ipa.test]]] [sysdb_sudo_purge_byfilter] (0x0400): No rules matched
(Thu Jun 21 10:37:47 2012) [sssd[be[ipa.test]]] [sysdb_sudo_purge_bysudouser] (0x0400): No rules matched
(Thu Jun 21 10:37:47 2012) [sssd[be[ipa.test]]] [sdap_sudo_load_sudoers_done] (0x0400): Sudoers is successfully stored in cache
(Thu Jun 21 10:37:47 2012) [sssd[be[ipa.test]]] [be_sudo_handler_reply] (0x0200): SUDO Backend returned: (0, 0, Success)
```

When using an Identity Management provider for SSSD, SSSD attempts to connect to the underlying LDAP directory using Kerberos/GSS-API. However, by default, SSSD uses an anonymous connection to an LDAP server to retrieve sudo rules. This means that SSSD cannot retrieve the sudo rules from the Identity Management server with its default configuration.

To support retrieving sudo rules with a Kerberos/GSS-API connection, enable GSS-API as the authentication mechanism in the identity provider configuration in **sssd.conf**. For example:

```
[domain/ipa.example.com]
id_provider = ipa
ipa_server = ipa.example.com
ldap_tls_cacert = /etc/ipa/ca.crt

sudo_provider = ldap
ldap_uri = ldap://ipa.example.com
ldap_sudo_search_base = ou=sudoers,dc=ipa,dc=example,dc=com
ldap_sasl_mech = GSSAPI
ldap_sasl_authid = host/hostname.ipa.example.com
ldap_sasl_realm = IPA.EXAMPLE.COM
krb5_server = ipa.example.com
```

Q: Password lookups on large directories can take several seconds per request. How can this be improved?

A: The initial user lookup is a call to the LDAP server. Unindexed searches are much more resource-intensive, and therefore take longer, than indexed searches because the server checks every entry in the directory for a match. To speed up user lookups, index the attributes that are searched for by SSSD:

uid

uidNumber

gidNumber

gecos

Q: An Active Directory identity provider is properly configured in my **sssd.conf file, but SSSD fails to connect to it, with GSS-API errors.**

A: SSSD can only connect with an Active Directory provider using its host name. If the host name is not given, the SSSD client cannot resolve the IP address to the host, and authentication fails.

For example, with this configuration:

```
[domain/ADEXAMPLE]
debug_level = 0xFFFF0
id_provider = ad
ad_server = 255.255.255.255
ad_domain = example.com
krb5_canonicalize = False
```

The SSSD client returns this GSS-API failure, and the authentication request fails:

```
(Fri Jul 27 18:27:44 2012) [sssd[be[ADTEST]]] [sasl_bind_send] (0x0020): ldap_sasl_bind failed
(-2)[Local error]
(Fri Jul 27 18:27:44 2012) [sssd[be[ADTEST]]] [sasl_bind_send] (0x0080): Extended failure
message: [SASL(-1): generic failure: GSSAPI Error: Unspecified GSS failure. Minor code may
provide more information (Cannot determine realm for numeric host address)]
```

To avoid this error, set the ***ad_server*** to the name of the Active Directory host.

Q: I configured SSSD for central authentication, but now several of my applications (such as Firefox or Adobe) will not start.

A: Even on 64-bit systems, 32-bit applications require a 32-bit version of SSSD to use to access the password and identity cache. If a 32-bit version of SSSD is not available, but the system is configured to use the SSSD cache, then 32-bit applications can fail to start.

For example, Firefox can fail with permission denied errors:

```
Failed to contact configuration server. See http://www.gnome.org/projects/gconf/
for information. (Details - 1: IOR file '/tmp/gconfd-somebody/lock/ior'
not opened successfully, no gconfd located: Permission denied 2: IOR
file '/tmp/gconfd-somebody/lock/ior' not opened successfully, no gconfd
located: Permission denied)
```

For Adobe Reader, the error shows that the current system user is not recognized:

```
~]$ acroread
(acroread:12739): GLib-WARNING **: getpwuid_r(): failed due to unknown
user id (366)
```

Other applications may show similar user or permissions errors.

Q: SSSD is showing an automount location that I removed.

A: The SSSD cache for the automount location persists even if the location is subsequently changed or removed. To update the autofs information in SSSD:

1. Remove the autofs cache, as described in [the section called “Purging the SSSD Cache”](#).
2. Restart SSSD, as in [Section 13.2.3, “Starting and Stopping SSSD”](#).

[3] Most services default to the identity provider server if a specific server for that service is not set.

CHAPTER 14. 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.

Red Hat Enterprise Linux includes the general OpenSSH package, **openssh**, as well as the OpenSSH server, **openssh-server**, and client, **openssh-clients**, packages.

14.1. THE SSH PROTOCOL

14.1.1. 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.

14.1.2. Main Features

The SSH protocol provides the following safeguards:

No one can pose as the intended server

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

No one can capture the authentication information

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

No one can intercept the communication

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

Additionally, it also offers the following options:

It provides secure means to use graphical applications over a network

Using a technique called *X11 forwarding*, the client can forward *X Window System* applications from the server. 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.

14.1.3. Protocol Versions

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



IMPORTANT

For maximum security, avoid using SSH version 1 and use SSH version 2-compatible servers and clients whenever possible.

14.1.4. 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.

14.1.4.1. 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.



WARNING

Always verify the integrity of a new SSH server. During the initial contact, an attacker can pretend to be the intended SSH server to the local system without being recognized. To verify the integrity of a new SSH server, contact the server administrator before the first connection or if a host key mismatch occurs.

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.

14.1.4.2. 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.

14.1.4.3. Channels

After a successful authentication over the SSH transport layer, multiple channels are opened via a technique called *multiplexing*^[4]. 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.

14.2. CONFIGURING OPENSSH

14.2.1. 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 14.1, "System-wide configuration files"](#). User-specific SSH configuration information is stored in **~/.ssh/** within the user's home directory as described in [Table 14.2, "User-specific configuration files"](#).

Table 14.1. System-wide configuration files

File	Description
------	-------------

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

Table 14.2. User-specific configuration files

File	Description
~/.ssh/authorized_keys	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.
~/.ssh/id_dsa	Contains the DSA private key of the user.
~/.ssh/id_dsa.pub	The DSA public key of the user.
~/.ssh/id_rsa	The RSA private key used by ssh for version 2 of the SSH protocol.

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

14.2.2. Starting an OpenSSH Server

In order to run an OpenSSH server, you must have the `openssh-server` installed (see [Section 8.2.4, “Installing Packages”](#) for more information on how to install new packages in Red Hat Enterprise Linux 6).

To start the **sshd** daemon, type the following at a shell prompt:

```
~]# service sshd start
```

To stop the running **sshd** daemon, use the following command:

```
~1# service sshd stop
```

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

```
~1# chkconfig sshd on
```

This will enable the service for levels 2, 3, 4, and 5. For more configuration options, see [Chapter 12, Services and Daemons](#) for the detailed information on how to manage services.

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:

To prevent this, you can backup the relevant files from the `/etc/ssh` directory (see [Table 14.1, “System-wide configuration files”](#) for a complete list), and restore them whenever you reinstall the system.

14.2.3. 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**.

To disable these services, type the following commands at a shell prompt:

```
~]# chkconfig telnet off
~]# chkconfig rsh off
~]# chkconfig rlogin off
~]# chkconfig vsftpd off
```

For more information on runlevels and configuring services in general, see [Chapter 12, Services and Daemons](#).

14.2.4. Using Key-Based Authentication

To improve the system security even further, you can enforce key-based authentication by disabling the standard 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
```

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

Red Hat Enterprise Linux 6 uses SSH Protocol 2 and RSA keys by default (see [Section 14.1.3, “Protocol Versions”](#) for more information).



IMPORTANT

Do not generate key pairs as **root**, as only **root** would be able to use those keys.



NOTE

Before reinstalling your system, back up the **~/.ssh/** directory to keep the generated key pair. Copy the backed-up data to the home directory in the new system for any user you require, including root.

14.2.4.1. Generating Key Pairs

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

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

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

2. Press **Enter** to confirm the default location (that is, **~/.ssh/id_rsa**) for the newly created key.

3. Enter a passphrase, and confirm it by entering it again when prompted to do so. For security reasons, avoid using the same password as you use to log in to your account.

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

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

4. Change the permissions of the `~/.ssh/` directory:

```
~]$ chmod 700 ~/.ssh
```

5. Copy the content of `~/.ssh/id_rsa.pub` into the `~/.ssh/authorized_keys` on the machine to which you want to connect, appending it to its end if the file already exists.
6. Change the permissions of the `~/.ssh/authorized_keys` file using the following command:

```
~]$ chmod 600 ~/.ssh/authorized_keys
```

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

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

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

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

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

```
Your identification has been saved in /home/john/.ssh/id_dsa.
Your public key has been saved in /home/john/.ssh/id_dsa.pub.
The key fingerprint is:
81:a1:91:a8:9f:e8:c5:66:0d:54:f5:90:cc:bc:cc:27 john@penguin.example.com
The key's randomart image is:
+--[ DSA 1024]----+
```

.oo*	o.
...o	Bo
... +	o.
. .	E o
o..o	S
. o = .	
. +	
.	
+-----+	

- #### 4. Change the permissions of the `~/.ssh/` directory:

```
~]$ chmod 700 ~/.ssh
```

5. Copy the content of `~/.ssh/id_dsa.pub` into the `~/.ssh/authorized_keys` on the machine to which you want to connect, appending it to its end if the file already exists.
 6. Change the permissions of the `~/.ssh/authorized_keys` file using the following command:

```
~]$ chmod 600 ~/.ssh/authorized_keys
```

To generate an RSA key pair for version 1 of the SSH protocol, follow these steps:

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

```
~]$ ssh-keygen -t rsa1
Generating public/private rsa1 key pair.
Enter file in which to save the key (/home/john/.ssh/identity):
```

2. Press **Enter** to confirm the default location (that is, `~/.ssh/identity`) 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 into your account.

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

```
Your identification has been saved in /home/john/.ssh/identity.
Your public key has been saved in /home/john/.ssh/identity.pub.
The key fingerprint is:
cb:f6:d5:cb:6e:5f:2b:28:ac:17:0c:e4:62:e4:6f:59 john@penguin.example.com
The key's randomart image is:
+--[RSA1 2048]----+
| |
| |
| .. |
| o o |
| + o E |
| . o S |
| = + . |
| . = . o ..|
| . = o o..o|
| .o o o=o.|
+-----+
```

4. Change the permissions of the `~/.ssh/` directory:

```
~]$ chmod 700 ~/.ssh
```

5. Copy the content of `~/.ssh/identity.pub` into the `~/.ssh/authorized_keys` on the machine to which you want to connect, appending it to its end if the file already exists.

6. Change the permissions of the `~/.ssh/authorized_keys` file using the following command:

```
~]$ chmod 600 ~/.ssh/authorized_keys
```

See [Section 14.2.4.2, “Configuring ssh-agent”](#) for information on how to set up your system to remember the passphrase.



IMPORTANT

Never share your private key with anybody; it is for your personal use only.

14.2.4.2. Configuring ssh-agent

To store your passphrase so that you do not have to enter it each time you initiate a connection with a remote machine, you can use the **ssh-agent** authentication agent. If you are running GNOME, you can configure it to prompt you for your passphrase whenever you log in and remember it during the whole session. Otherwise you can store the passphrase for a certain shell prompt.

To save your passphrase during your GNOME session, follow these steps:

1. Make sure you have the `openssh-askpass` package installed. If not, see [Section 8.2.4, “Installing Packages”](#) for more information on how to install new packages in Red Hat Enterprise Linux.
2. Select **System → Preferences → Startup Applications** from the panel. The **Startup Applications Preferences** will be started, and the tab containing a list of available startup programs will be shown by default.

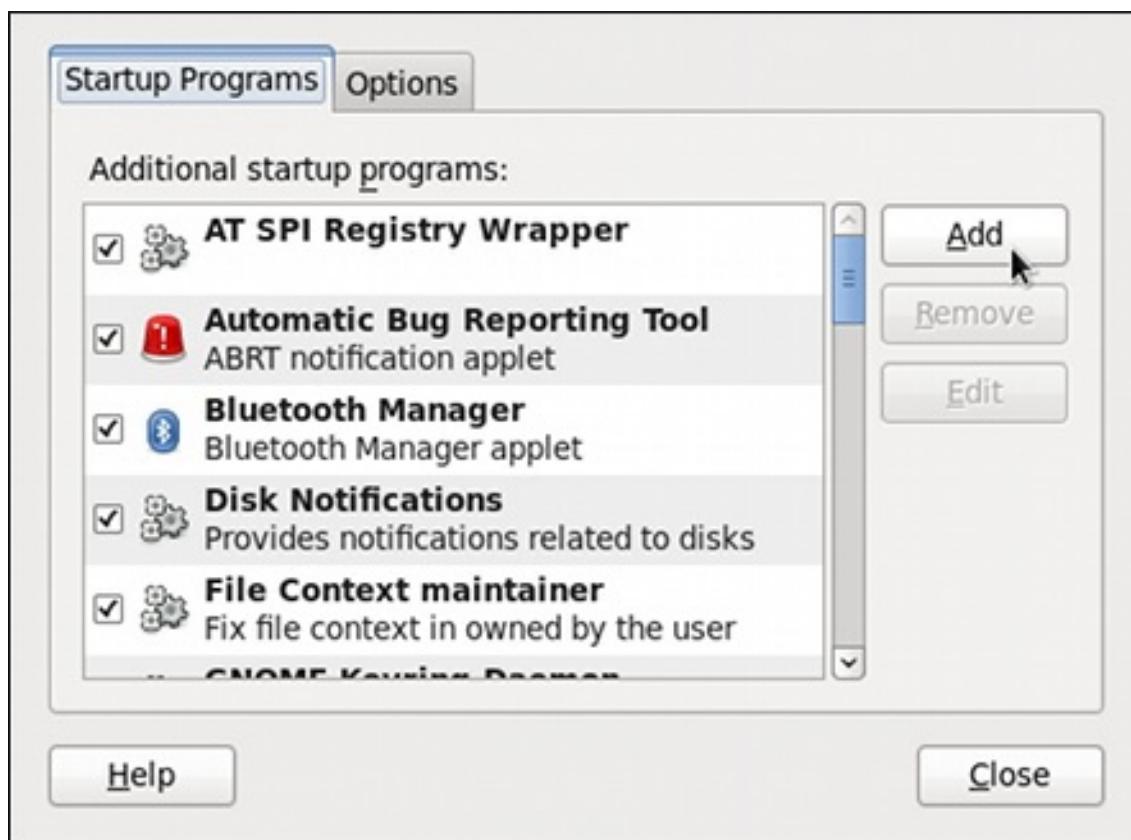


Figure 14.1. Startup Applications Preferences

3. Click the **Add** button on the right, and enter `/usr/bin/ssh-add` in the **Command** field.

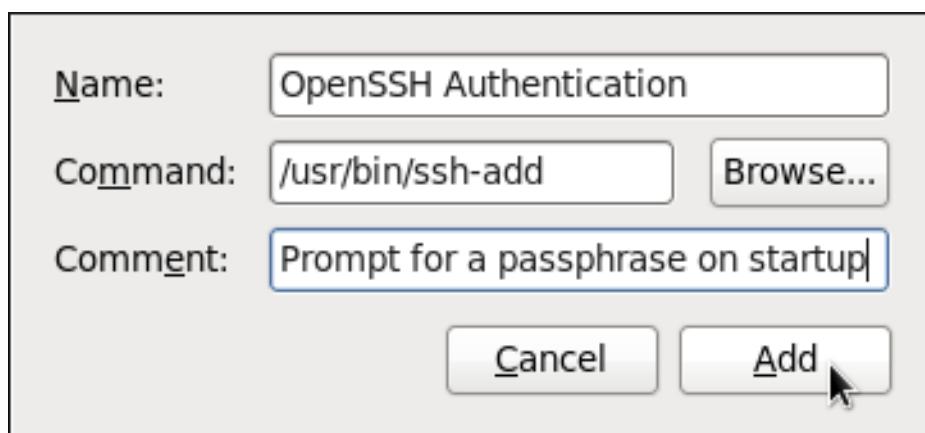


Figure 14.2. Adding new application

4. Click **Add** and make sure the check box next to the newly added item is selected.

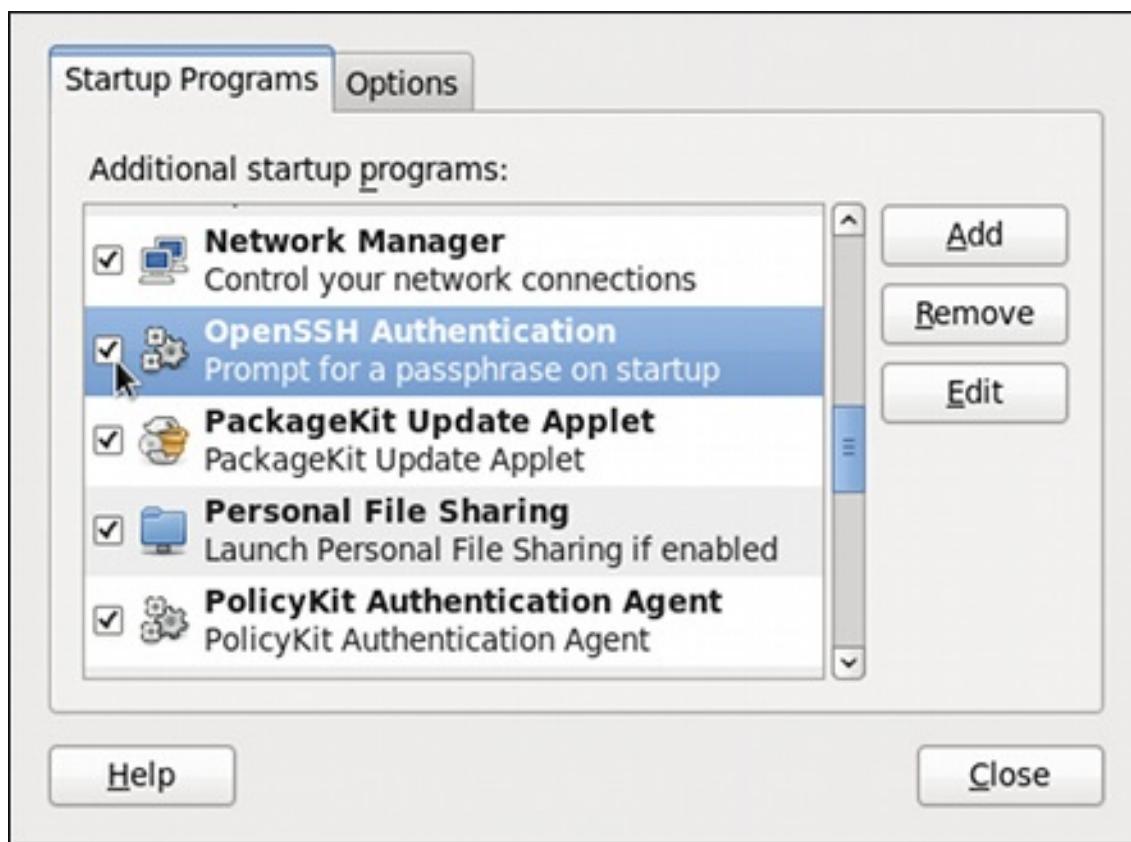


Figure 14.3. Enabling the application

5. Log out and then log back in. A dialog box will appear prompting you for your passphrase. From this point on, you should not be prompted for a password by **ssh**, **scp**, or **sftp**.

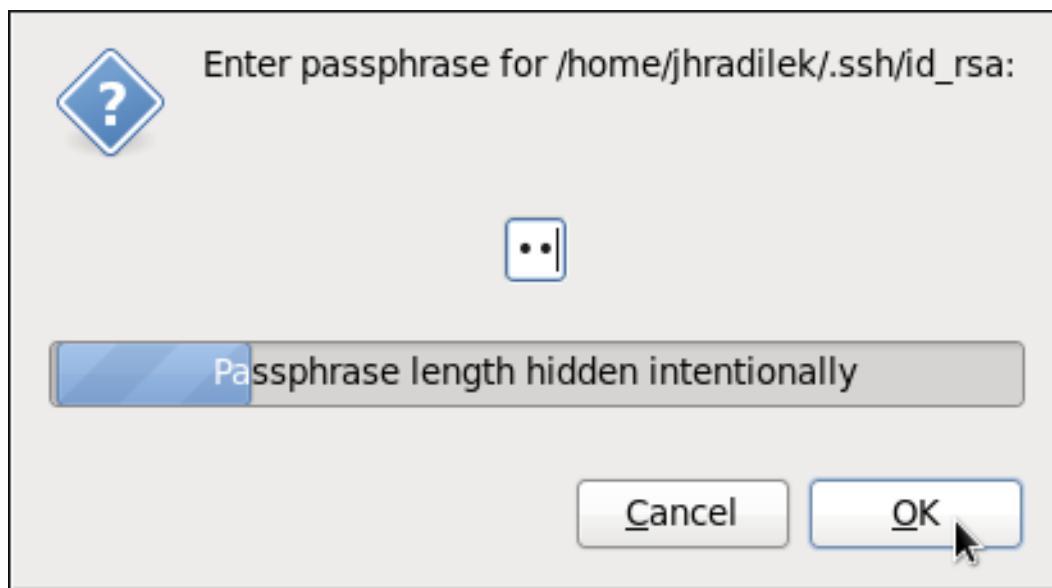


Figure 14.4. Entering a passphrase

To save your passphrase for a certain shell prompt, use the following command:

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

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

14.2.4.3. Multiple required methods of authentication for sshd

For higher security, SSH can require multiple methods of authentication to log in successfully, for example both a passphrase and a public key. Set the **RequiredAuthentications2** option in the **/etc/ssh/sshd_config** file as desired, for example by running:

```
~]# echo "RequiredAuthentications2 publickey,password" >> /etc/ssh/sshd_config
```

For more information on the available options, see the **sshd_config(5)** manual page.

14.3. USING OPENSSH CERTIFICATE AUTHENTICATION

14.3.1. 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.

14.3.2. Support for SSH Certificates

Support for certificate authentication of users and hosts using the new OpenSSH certificate format was introduced in Red Hat Enterprise Linux 6.5, in the **openssh-5.3p1-94.el6** package. If required, to ensure the latest OpenSSH package is installed, enter the following command as **root**:

```
~]# yum install openssh
Package openssh-5.3p1-104.el6_6.1.i686 already installed and latest version
Nothing to do
```

14.3.3. 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 -l certificate_ID id_rsa.pub
```

Where **-s** indicates the private key used to sign the certificate, **-l** 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 -l 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](#).

Procedure 14.1. 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]---+
|   .+. o|
|   . 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.. |
| *.E. |
| +o= |
| .oo. |
+-----+
```

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 -l certificate_ID -h -Z host_name.example.com -V -
start:+end /etc/ssh/ssh_host_rsa.pub
```

The `-Z` option restricts this certificate to a specific host within the domain. The `-V` option is for adding a validity period; this is highly recommended. 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 -l host_name -h -Z 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
```

14.3.4. 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 14.2. 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 14.3. 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
```

-
2. 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.

14.3.5. 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 ..
```

14.3.5.1. Creating SSH Certificates to Authenticate Hosts

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

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

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

Procedure 14.4. 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 -l host_name -h -Z host_name.example.com -V -
1d:+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 14.3, "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.

14.3.5.2. 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 -l user_name -Z 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 **-Z** option:


```
-Z "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 14.5. 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]----+
| 00++.   |
| 0.0.0.   |
| .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 14.2.4, “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 14.3, “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 -l user1 -Z user1 -V -1d:+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.

14.3.6. 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 -l 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 **-Z** option as follows:

```
ssh-keygen -s ca_user_key.pub -D libpkcs11.so -l certificate_ID -Z user1,user2 id_rsa.pub
```

and for hosts:

```
ssh-keygen -s ca_host_key.pub -D libpkcs11.so -l certificate_ID -Z 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 -l 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.

14.3.7. 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)

```

14.3.8. 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 than 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.

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.

14.4. OPENSSH CLIENTS

To connect to an OpenSSH server from a client machine, you must have the `openssh-clients` and `openssh` packages installed (see [Section 8.2.4, “Installing Packages”](#) for more information on how to install new packages in Red Hat Enterprise Linux).

14.4.1. 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 **john**, type:

```
~]$ ssh john@penguin.example.com
```

The first time you initiate a connection, you will be presented with a message similar to this:

```
The authenticity of host 'penguin.example.com' can't be established.  
RSA key fingerprint is 94:68:3a:3a:bc:f3:9a:9b:01:5d:b3:07:38:e2:11:0c.  
Are you sure you want to continue connecting (yes/no)?
```

Type **yes** to confirm. 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' (RSA) to the list of known hosts.  
john@penguin.example.com's password:
```



IMPORTANT

Update the host key of an SSH server if the 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. Contact the system administrator of the SSH server to verify the server is not compromised, then remove the line with the name of the remote machine at the beginning.

After entering the password, you will be provided with a shell prompt for the remote machine.

Alternatively, the **ssh** program can be used to execute a command on the remote machine without logging in to a shell prompt:

```
ssh [username@]hostname command
```

For example, the **/etc/redhat-release** file provides information about the Red Hat Enterprise Linux version. To view the contents of this file on **penguin.example.com**, type:

```
~]$ ssh john@penguin.example.com cat /etc/redhat-release  
john@penguin.example.com's password:  
Red Hat Enterprise Linux Server release 6.2 (Santiago)
```

After you enter the correct password, the user name will be displayed, and you will return to your local shell prompt.

14.4.2. 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 john@penguin.example.com:.vim/plugin/taglist.vim
john@penguin.example.com's password:
taglist.vim          100% 144KB 144.5KB/s 00:00
```

Multiple files can be specified at once. To transfer the contents of **.vim/plugin/** to the same directory on the remote machine **penguin.example.com**, type the following command:

```
~]$ scp .vim/plugin/* john@penguin.example.com:.vim/plugin/
john@penguin.example.com's password:
closetag.vim           100% 13KB 12.6KB/s 00:00
snippetsEmu.vim         100% 33KB 33.1KB/s 00:00
taglist.vim            100% 144KB 144.5KB/s 00:00
```

To transfer a remote file to the local system, use the following syntax:

```
scp username@hostname:remotefile localfile
```

For instance, to download the **.vimrc** configuration file from the remote machine, type:

```
~]$ scp john@penguin.example.com:.vimrc .vimrc
john@penguin.example.com's password:
.vimrc                100% 2233  2.2KB/s 00:00
```

14.4.3. 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 **john** as a user name, type:

```
~]$ sftp john@penguin.example.com
john@penguin.example.com's password:
Connected to penguin.example.com.
sftp>
```

After you enter the correct password, you will be presented with a prompt. The **sftp** utility accepts a set of commands similar to those used by **ftp** (see [Table 14.3, "A selection of available sftp commands"](#)).

Table 14.3. 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> .

Command	Description
mkdir <i>directory</i>	Create a remote <i>directory</i> .
rmdir <i>path</i>	Remove a remote <i>directory</i> .
put <i>localfile</i> [<i>remotefile</i>]	Transfer <i>localfile</i> to a remote machine.
get <i>remotefile</i> [<i>localfile</i>]	Transfer <i>remotefile</i> from a remote machine.

For a complete list of available commands, see the **sftp(1)** manual page.

14.5. 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.

14.5.1. 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 **john** as a user name, type:

```
~]$ ssh -Y john@penguin.example.com
john@penguin.example.com's password:
```

When an X program is run from the secure shell prompt, the SSH client and server create a new secure channel, and the X program data is sent over that channel to the client machine transparently.

X11 forwarding can be very useful. For example, X11 forwarding can be used to create a secure, interactive session of the **Printer Configuration** utility. To do this, connect to the server using **ssh** and type:

```
~]$ system-config-printer &
```

The **Printer Configuration Tool** will appear, allowing the remote user to safely configure printing on the remote system.

Please note that X11 Forwarding does not distinguish between trusted and untrusted forwarding.

14.5.2. 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.



NOTE

If you want to use reserved port numbers, please note that setting up port forwarding to listen on ports below 1024 requires **root** level access.

To create a TCP/IP port forwarding channel which listens for connections on the **localhost**, use a command in the following form:

```
ssh -L local-port:remote-hostname:remote-port username@hostname
```

For example, to check email on a server called **mail.example.com** using **POP3** through an encrypted connection, use the following command:

```
~]$ ssh -L 1100:mail.example.com:110 mail.example.com
```

Once the port forwarding channel is in place between the client machine and the mail server, direct a POP3 mail client to use port **1100** on the **localhost** to check for new email. Any requests sent to port **1100** on the client system will be directed securely to the **mail.example.com** server.

If **mail.example.com** is not running an SSH server, but another machine on the same network is, SSH can still be used to secure part of the connection. However, a slightly different command is necessary:

```
~]$ ssh -L 1100:mail.example.com:110 other.example.com
```

In this example, POP3 requests from port **1100** on the client machine are forwarded through the SSH connection on port **22** to the SSH server, **other.example.com**. Then, **other.example.com** connects to port **110** on **mail.example.com** to check for new email. Note that when using this technique, only the connection between the client system and **other.example.com** SSH server is secure.

Port forwarding can also be used to get information securely through network firewalls. If the firewall is configured to allow SSH traffic via its standard port (that is, port 22) but blocks access to other ports, a connection between two hosts using the blocked ports is still possible by redirecting their communication over an established SSH connection.



IMPORTANT

The connection is only as secure as the client system because forwarding connections in this way allows any user on the client system to connect to that service. If the client system becomes compromised, an attacker can also access the forwarded services.

If preferred, disable this functionality on the server by specifying a **No** parameter for the **AllowTcpForwarding** line in the **/etc/ssh/sshd_config** file and restarting the **sshd** service.

14.6. ADDITIONAL RESOURCES

For more information about OpenSSH and OpenSSL, see the resources listed below.

14.6.1. Installed Documentation

- **sshd(8)** – a manual page for the **sshd** daemon.
- **ssh(1)** – a manual page for the **ssh** client.
- **scp(1)** – a manual page for the **scp** utility.
- **sftp(1)** – a manual page for the **sftp** utility.
- **ssh-keygen(1)** – a manual page for the **ssh-keygen** utility.
- **ssh_config(5)** – a manual page with a full description of available SSH client configuration options.
- **sshd_config(5)** – a manual page with a full description of available SSH daemon configuration options.
- **/usr/share/doc/openssh-version/** Contains detailed information on the protocols supported by OpenSSH.

14.6.2. Useful Websites

<http://www.openssh.com/>

The OpenSSH home page containing further documentation, frequently asked questions, links to the mailing lists, bug reports, and other useful resources.

<http://www.openssl.org/>

The OpenSSL home page containing further documentation, frequently asked questions, links to the mailing lists, and other useful resources.

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

CHAPTER 15. 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 principle: a **server** shares its output (**vncserver**) and a **client** (**vncviewer**) connects to the server.

15.1. 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.

15.1.1. Installing VNC Server

To install the **TigerVNC** server, run the following command as **root**:

```
~]# yum install tigervnc-server
```

15.1.2. Configuring VNC Server

The VNC server can be configured to start a display for one or more users, provided that accounts for the users exist on the system, with optional parameters such as for display settings, network address and port, and security settings.

Procedure 15.1. Configuring a VNC Display for a Single User

- Specify the user name and the display number by editing **/etc/sysconfig/vncservers** and adding a line in the following format:

```
VNCSEVERS="display_number:user"
```

The VNC user names must correspond to users of the system.

Example 15.1. Setting the Display Number for a User

For example, to configure display number **3** for user **joe**, open the configuration file for editing:

```
~]# vi /etc/sysconfig/vncservers
```

Add a line as follows:

```
VNCSEVERS="3:joe"
```

Save and close the file.

In the example above, display number **3** and the user **joe** are set. Do not use **0** as the display number since the main X display of a workstation is usually indicated as **0**.

Procedure 15.2. Configuring a VNC Display for Multiple Users

- To set a VNC display for more than one user, specify the user names and display numbers by editing **/etc/sysconfig/vncservers** and adding a line in the following format:

```
VNCERVERS="display_number:user display_number:user"
```

The VNC user names must correspond to users of the system.

Example 15.2. Setting the Display Numbers for Two Users

For example, to configure two users, open the configuration file for editing:

```
~]# vi /etc/sysconfig/vncservers
```

Add a line as follows:

```
VNCERVERS="3:joe 4:jill"
```

Procedure 15.3. Configuring VNC Display Arguments

- Specify additional settings in the **/etc/sysconfig/vncservers** file by adding arguments using the VNCERVERARGS directive as follows:

```
VNCERVERS="display_number:user display_number:user"
VNCERVERARGS[display_number]="arguments"
```

Table 15.1. Frequently Used VNC Server Parameters

VNCERVERARGS	Definition
-geometry	specifies the size of the VNC desktop to be created, default is 1024x768.
-nolisten tcp	prevents connections to your VNC server through TCP (Transmission Control Protocol)
-localhost	prevents remote VNC clients from connecting except when doing so through a secure tunnel

See the **Xvnc(1)** man page for further options.

Example 15.3. Setting vncserver Arguments

Following on from the example above, to add arguments for two users, edit the **/etc/sysconfig/vncservers** file as follows:

```
VNCERVERS="3:joe 4:jill"
VNCERVERARGS[1]="-geometry 800x600 -nolisten tcp -localhost"
VNCERVERARGS[2]="-geometry 1920×1080 -nolisten tcp -localhost"
```

Procedure 15.4. Configuring VNC User Passwords

- To set the VNC password for all users defined in the **/etc/sysconfig/vncservers** file, enter the following command as **root**:

```
~]# vncpasswd
Password:
Verify:
```

To set the VNC password individually for a user:

```
~]# 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.

15.1.3. Starting VNC Server

In order to start a VNC desktop, the **vncserver** utility is used. It is a Perl script which simplifies the process of starting an Xvnc server. It runs Xvnc with appropriate options and starts a window manager on the VNC desktop. There are three ways to start **vncserver**:

- You can allow **vncserver** to choose the first available display number, start Xvnc with that display number, and start the default window manager in the Xvnc session. All these steps are provided by one command:

```
~]$ vncserver
```

You will be prompted to enter a VNC password the first time the command is run if no VNC password has been set.

- Alternately, you can specify a specific display number:

```
vncserver :display_number
```

vncserver attempts to start Xvnc with that display number and exits if the display number is not available.

For example:

```
~]$ vncserver :20
```

- Alternately, to start VNC server with displays for the users configured in the **/etc/sysconfig/vncservers** configuration file, as **root** enter:

```
~]# service vncserver start
```

You can enable the **vncserver** service automatically at system start. Every time you log in, **vncserver** is automatically started. As **root**, run

```
~]# chkconfig vncserver on
```

15.1.4. Terminating a VNC Session

Similarly to enabling the **vncserver** service, you can disable the automatic start of the service at system start:

```
~]# chkconfig vncserver off
```

Or, when your system is running, you can stop the service by issuing the following command as **root**:

```
~]# service vncserver stop
```

To terminate a specific display, terminate **vncserver** using the **-kill** option along with the display number.

Example 15.4. Terminating a Specific Display

For example, to terminate display number 2, run:

```
~]# vncserver -kill :2
```

Example 15.5. Terminating an Xvnc process

If it is not possible to terminate the VNC service or display, terminate the Xvnc session using the process ID (PID). To view the processes, enter:

```
~]$ service vncserver status
Xvnc (pid 4290 4189) is running...
```

To terminate process **4290**, enter as **root**:

```
~]# kill -s 15 4290
```

15.2. SHARING AN EXISTING DESKTOP

By default a logged in user has a desktop provided by X Server on display **0**. A user can share their desktop using the **TigerVNC** server **x0vncserver**.

Procedure 15.5. Sharing an X Desktop

To share the desktop of a logged in user, using the **x0vncserver**, proceed as follows:

1. Enter the following command as **root**

```
~]# yum install tigervnc-server
```

- Set the VNC password for the user:

```
~]$ vncpasswd
Password:
Verify:
```

- Enter the following command as that user:

```
~]$ x0vncserver -PasswordFile=.vnc/passwd -AlwaysShared=1
```

Provided the firewall is configured to allow connections to port **5900**, the remote viewer can now connect to display **0**, and view the logged in users desktop. See [Section 15.3.2.1, “Configuring the Firewall for VNC”](#) for information on how to configure the firewall.

15.3. USING A VNC VIEWER

A VNC viewer is a program which shows the graphical user interface created by the VNC server and can control the VNC server remotely. The desktop that is shared is not by default the same as the desktop that is displayed to a user directly logged into the system. The VNC server creates a unique desktop for every display number. Any number of clients can connect to a VNC server.

15.3.1. Installing the VNC Viewer

To install the TigerVNC client, **vncviewer**, as **root**, run the following command:

```
~]# yum install tigervnc
```

The **TigerVNC** client has a graphical user interface (GUI) which can be started by entering the command **vncviewer**. Alternatively, you can operate **vncviewer** through the command-line interface (CLI). To view a list of parameters for **vncviewer** enter **vncviewer -h** on the command line.

15.3.2. Connecting to a VNC Server

Once the VNC server is configured, you can connect to it from any VNC viewer.

Procedure 15.6. Connecting to a VNC Server Using a GUI

- Enter the **vncviewer** command with no arguments, the **VNC Viewer: Connection Details** utility appears. It prompts for a VNC server to connect to.
- If required, to prevent disconnecting any existing VNC connections to the same display, select the option to allow sharing of the desktop as follows:
 - Select the **Options** button.
 - Select the **Misc.** tab.
 - Select the **Shared** button.
 - Press **OK** to return to the main menu.
- Enter an address and display number to connect to:

```
address:display_number
```

-
- 4. Press **Connect** to connect to the VNC server display.
- 5. You will be prompted to enter the VNC password. This will be the VNC password for the user corresponding to the display number unless a global default VNC password was set.

A window appears showing the VNC server desktop. Note that this is not the desktop the normal user sees, it is an Xvnc desktop.

Procedure 15.7. Connecting to a VNC Server Using the CLI

1. Enter the **viewer** command with the address and display number as arguments:

```
vncviewer address:display_number
```

Where *address* is an **IP** address or host name.

2. Authenticate yourself by entering the VNC password. This will be the VNC password for the user corresponding to the display number unless a global default VNC password was set.
3. A window appears showing the VNC server desktop. Note that this is not the desktop the normal user sees, it is the Xvnc desktop.

15.3.2.1. Configuring the Firewall for VNC

When using a non-encrypted connection, the firewall might block your connection. The VNC protocol is *remote framebuffer* (RFB), which is transported in **TCP** packets. If required, open a port for the **TCP** protocol as described below. When using the **-via** option, traffic is redirected over **SSH** which is enabled by default.



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 display: $2 + 5900 = 5902$.

Procedure 15.8. Opening a Port Using lokkit

The **lokkit** command provides a way to quickly enable a port using the command line.

1. To enable a specific port, for example port **5902** for **TCP**, issue the following command as **root**:

```
~]# lokkit --port=5902:tcp --update
```

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

2. Verify whether the chosen port is open. As **root**, enter:

```
~]# iptables -L -n | grep 'tcp.*59'
ACCEPT  tcp  --  0.0.0.0/0  0.0.0.0/0  state NEW tcp dpt:5902
```

3. If you are unsure of the port numbers in use for VNC, as **root**, enter:

```
~]# netstat -tnlp
tcp  0  0 0.0.0.0:6003  0.0.0.0:*  LISTEN  4290/Xvnc
tcp  0  0 0.0.0.0:5900  0.0.0.0:*  LISTEN  7013/x0vncserver
tcp  0  0 0.0.0.0:5902  0.0.0.0:*  LISTEN  4189/Xvnc
tcp  0  0 0.0.0.0:5903  0.0.0.0:*  LISTEN  4290/Xvnc
tcp  0  0 0.0.0.0:6002  0.0.0.0:*  LISTEN  4189/Xvnc
```

Ports starting **59XX** are for the VNC **RFB** protocol. Ports starting **60XX** are for the X windows protocol.

To list the ports and the Xvnc session's associated user, as **root**, enter:

```
~]# lsof -i -P | grep vnc
Xvnc  4189  jane  0u  IPv6  27972  0t0  TCP *:6002 (LISTEN)
Xvnc  4189  jane  1u  IPv4  27973  0t0  TCP *:6002 (LISTEN)
Xvnc  4189  jane  6u  IPv4  27979  0t0  TCP *:5902 (LISTEN)
Xvnc  4290  joe   0u  IPv6  28231  0t0  TCP *:6003 (LISTEN)
Xvnc  4290  joe   1u  IPv4  28232  0t0  TCP *:6003 (LISTEN)
Xvnc  4290  joe   6u  IPv4  28244  0t0  TCP *:5903 (LISTEN)
x0vncserv 7013  joe  4u  IPv4  47578  0t0  TCP *:5900 (LISTEN)
```

Procedure 15.9. Configuring the Firewall Using an Editor

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

1. To check what the firewall is configured to allow, issue the following command as **root** to view the firewall configuration file:

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

--enabled
--service=ssh
```

In this example taken from a default installation, the firewall is enabled but VNC ports have not been configured to pass through.

2. Open **/etc/sysconfig/system-config-firewall** for editing as **root** and add lines in the following format to the firewall configuration file:

```
--port=port_number:tcp
```

For example, to add port **5902**:

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

--enabled
--service=ssh
--port=5902:tcp
```

3. Note that these changes will not take effect even if the firewall is reloaded or the system rebooted. To apply the settings in **/etc/sysconfig/system-config-firewall**, issue the following command as **root**:

```
~]# lokkit --update
```

15.3.3. 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 15.6. Using the -via Option

1. To connect to a VNC server using **SSH**, enter a command as follows:

```
$ vncviewer -via joe@192.168.2.101 127.0.0.1:3
```

2. When you are prompted to, type the password, and confirm by pressing **Enter**.
3. A window with a remote desktop appears on your screen.

For more information on using **SSH**, see [Chapter 14, OpenSSH](#).

15.4. ADDITIONAL RESOURCES

For more information about TigerVNC, see the resources listed below.

Installed Documentation

- **vncserver(1)** – The manual page for the VNC server utility.
- **vncviewer(1)** – The manual page for the VNC viewer.
- **vncpasswd(1)** – The manual page for the VNC password command.
- **Xvnc(1)** – The manual page for the Xvnc server configuration options.
- **x0vncserver(1)** – The manual page for the **TigerVNC** server for sharing existing X servers.

PART VI. 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 16. DHCP SERVERS

Dynamic Host Configuration Protocol (DHCP) is a network protocol that automatically assigns TCP/IP information to client machines. Each DHCP client connects to the centrally located DHCP server, which returns the network configuration (including the IP address, gateway, and DNS servers) of that client.

16.1. WHY USE DHCP?

DHCP is useful for automatic configuration of client network interfaces. When configuring the client system, you can choose DHCP instead of specifying an IP address, netmask, gateway, or DNS servers. The client retrieves this information from the DHCP server. DHCP is also useful if you want to change the IP addresses of a large number of systems. Instead of reconfiguring all the systems, you can just edit one configuration file on the server for the new set of IP addresses. If the DNS servers for an organization changes, the changes happen on the DHCP server, not on the DHCP clients. When you restart the network or reboot the clients, the changes go into effect.

If an organization has a functional DHCP server correctly connected to a network, laptops and other mobile computer users can move these devices from office to office.

16.2. CONFIGURING A DHCPV4 SERVER

The `dhcp` package contains an Internet Systems Consortium (ISC) DHCP server. First, install the package as the superuser:

```
~]# yum install dhcp
```

Installing the `dhcp` package creates a file, **/etc/dhcp/dhcpd.conf**, which is merely an empty configuration file:

```
~]# cat /etc/dhcp/dhcpd.conf
#
# DHCP Server Configuration file.
# see /usr/share/doc/dhcp*/dhcpd.conf.sample
```

The sample configuration file can be found at **/usr/share/doc/dhcp-<version>/dhcpd.conf.sample**. You should use this file to help you configure **/etc/dhcp/dhcpd.conf**, which is explained in detail below.

DHCP also uses the file **/var/lib/dhcpd/dhcpd.leases** to store the client lease database. See [Section 16.2.2, “Lease Database”](#) for more information.

16.2.1. Configuration File

The first step in configuring a DHCP server is to create the configuration file that stores the network information for the clients. Use this file to declare options and global options for client systems.

The configuration file can contain extra tabs or blank lines for easier formatting. Keywords are case-insensitive and lines beginning with a hash sign (#) are considered comments.

There are two types of statements in the configuration file:

- Parameters – State how to perform a task, whether to perform a task, or what network configuration options to send to the client.

- Declarations – Describe the topology of the network, describe the clients, provide addresses for the clients, or apply a group of parameters to a group of declarations.

The parameters that start with the keyword option are referred to as *options*. These options control DHCP options; whereas, parameters configure values that are not optional or control how the DHCP server behaves.

Parameters (including options) declared before a section enclosed in curly brackets ({}) are considered global parameters. Global parameters apply to all the sections below it.



IMPORTANT

If the configuration file is changed, the changes do not take effect until the DHCP daemon is restarted with the command **service dhcpcd restart**.



NOTE

Instead of changing a DHCP configuration file and restarting the service each time, using the **omshell** command provides an interactive way to connect to, query, and change the configuration of a DHCP server. By using **omshell**, all changes can be made while the server is running. For more information on **omshell**, see the **omshell** man page.

In [Example 16.1, “Subnet Declaration”](#), the **routers**, **subnet-mask**, **domain-search**, **domain-name-servers**, and **time-offset** options are used for any **host** statements declared below it.

For every **subnet** which will be served, and for every **subnet** to which the DHCP server is connected, there must be one **subnet** declaration, which tells the DHCP daemon how to recognize that an address is on that **subnet**. A **subnet** declaration is required for each **subnet** even if no addresses will be dynamically allocated to that **subnet**.

In this example, there are global options for every DHCP client in the subnet and a **range** declared. Clients are assigned an IP address within the **range**.

Example 16.1. Subnet Declaration

```
subnet 192.168.1.0 netmask 255.255.255.0 {
    option routers          192.168.1.254;
    option subnet-mask       255.255.255.0;
    option domain-search     "example.com";
    option domain-name-servers 192.168.1.1;
    option time-offset      -18000;  # Eastern Standard Time
    range 192.168.1.10 192.168.1.100;
}
```

To configure a DHCP server that leases a dynamic IP address to a system within a subnet, modify [Example 16.2, “Range Parameter”](#) with your values. It declares a default lease time, maximum lease time, and network configuration values for the clients. This example assigns IP addresses in the **range** 192.168.1.10 and 192.168.1.100 to client systems.

Example 16.2. Range Parameter

```
default-lease-time 600;
max-lease-time 7200;
```

```

option subnet-mask 255.255.255.0;
option broadcast-address 192.168.1.255;
option routers 192.168.1.254;
option domain-name-servers 192.168.1.1, 192.168.1.2;
option domain-search "example.com";
subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.10 192.168.1.100;
}

```

To assign an IP address to a client based on the MAC address of the network interface card, use the **hardware ethernet** parameter within a **host** declaration. As demonstrated in [Example 16.3, "Static IP Address Using DHCP"](#), the **host apex** declaration specifies that the network interface card with the MAC address 00:A0:78:8E:9E:AA always receives the IP address 192.168.1.4.

Note that you can also use the optional parameter **host-name** to assign a host name to the client.

Example 16.3. Static IP Address Using DHCP

```

host apex {
    option host-name "apex.example.com";
    hardware ethernet 00:A0:78:8E:9E:AA;
    fixed-address 192.168.1.4;
}

```

All subnets that share the same physical network should be declared within a **shared-network** declaration as shown in [Example 16.4, "Shared-network Declaration"](#). Parameters within the **shared-network**, but outside the enclosed **subnet** declarations, are considered to be global parameters. The name of the **shared-network** must be a descriptive title for the network, such as using the title 'test-lab' to describe all the subnets in a test lab environment.

Example 16.4. Shared-network Declaration

```

shared-network name {
    option domain-search      "test.redhat.com";
    option domain-name-servers ns1.redhat.com, ns2.redhat.com;
    option routers            192.168.0.254;
    #more parameters for EXAMPLE shared-network
    subnet 192.168.1.0 netmask 255.255.252.0 {
        #parameters for subnet
        range 192.168.1.1 192.168.1.254;
    }
    subnet 192.168.2.0 netmask 255.255.252.0 {
        #parameters for subnet
        range 192.168.2.1 192.168.2.254;
    }
}

```

As demonstrated in [Example 16.5, "Group Declaration"](#), the **group** declaration is used to apply global parameters to a group of declarations. For example, shared networks, subnets, and hosts can be grouped.

Example 16.5. Group Declaration

```

group {
    option routers          192.168.1.254;
    option subnet-mask       255.255.255.0;
    option domain-search     "example.com";
    option domain-name-servers 192.168.1.1;
    option time-offset      -18000;  # Eastern Standard Time
    host apex {
        option host-name "apex.example.com";
        hardware ethernet 00:A0:78:8E:9E:AA;
        fixed-address 192.168.1.4;
    }
    host raleigh {
        option host-name "raleigh.example.com";
        hardware ethernet 00:A1:DD:74:C3:F2;
        fixed-address 192.168.1.6;
    }
}

```



NOTE

You can use the provided sample configuration file as a starting point and add custom configuration options to it. To copy this file to the proper location, use the following command as **root**:

```

~]# cp /usr/share/doc/dhcp-<version_number>/dhcpd.conf.sample
/etc/dhcp/dhcpd.conf

```

... where *<version_number>* is the DHCP version number.

For a complete list of option statements and what they do, see the **dhcp-options** man page.

16.2.2. Lease Database

On the DHCP server, the file **/var/lib/dhcpd/dhcpd.leases** stores the DHCP client lease database. Do not change this file. DHCP lease information for each recently assigned IP address is automatically stored in the lease database. The information includes the length of the lease, to whom the IP address has been assigned, the start and end dates for the lease, and the MAC address of the network interface card that was used to retrieve the lease.

All times in the lease database are in Coordinated Universal Time (UTC), not local time.

The lease database is recreated from time to time so that it is not too large. First, all known leases are saved in a temporary lease database. The **dhcpd.leases** file is renamed **dhcpd.leases~** and the temporary lease database is written to **dhcpd.leases**.

The DHCP daemon could be killed or the system could crash after the lease database has been renamed to the backup file but before the new file has been written. If this happens, the **dhcpd.leases** file does not exist, but it is required to start the service. Do not create a new lease file. If you do, all old leases are lost which causes many problems. The correct solution is to rename the **dhcpd.leases~** backup file to **dhcpd.leases** and then start the daemon.

16.2.3. Starting and Stopping the Server



IMPORTANT

When the DHCP server is started for the first time, it fails unless the **`dhcpd.leases`** file exists. Use the command **`touch /var/lib/dhcpd/dhcpd.leases`** to create the file if it does not exist.

If the same server is also running BIND as a DNS server, this step is not necessary, as starting the **`named`** service automatically checks for a **`dhcpd.leases`** file.

To start the DHCP service, use the command **`/sbin/service dhcpcd start`**. To stop the DHCP server, use the command **`/sbin/service dhcpcd stop`**.

By default, the DHCP service does not start at boot time. For information on how to configure the daemon to start automatically at boot time, see [Chapter 12, Services and Daemons](#).

If more than one network interface is attached to the system, but the DHCP server should only be started on one of the interfaces, configure the DHCP server to start only on that device. In **`/etc/sysconfig/dhcpcd`**, add the name of the interface to the list of **DHCPDARGS**:

```
# Command line options here
DHCPDARGS=eth0
```

This is useful for a firewall machine with two network cards. One network card can be configured as a DHCP client to retrieve an IP address to the Internet. The other network card can be used as a DHCP server for the internal network behind the firewall. Specifying only the network card connected to the internal network makes the system more secure because users cannot connect to the daemon through the Internet.

Other command-line options that can be specified in **`/etc/sysconfig/dhcpcd`** include:

- **`-p <portnum>`** – Specifies the UDP port number on which **`dhcpcd`** should listen. The default is port 67. The DHCP server transmits responses to the DHCP clients at a port number one greater than the UDP port specified. For example, if the default port 67 is used, the server listens on port 67 for requests and responds to the client on port 68. If a port is specified here and the DHCP relay agent is used, the same port on which the DHCP relay agent should listen must be specified. See [Section 16.2.4, “DHCP Relay Agent”](#) for details.
- **`-f`** – Runs the daemon as a foreground process. This is mostly used for debugging.
- **`-d`** – Logs the DHCP server daemon to the standard error descriptor. This is mostly used for debugging. If this is not specified, the log is written to **`/var/log/messages`**.
- **`-cf <filename>`** – Specifies the location of the configuration file. The default location is **`/etc/dhcpc/dhcpcd.conf`**.
- **`-lf <filename>`** – Specifies the location of the lease database file. If a lease database file already exists, it is very important that the same file be used every time the DHCP server is started. It is strongly recommended that this option only be used for debugging purposes on non-production machines. The default location is **`/var/lib/dhcpc/dhcpcd.leases`**.
- **`-q`** – Do not print the entire copyright message when starting the daemon.

16.2.4. DHCP Relay Agent

The DHCP Relay Agent (**dhcrelay**) allows for the relay of DHCP and BOOTP requests from a subnet with no DHCP server on it to one or more DHCP servers on other subnets.

When a DHCP client requests information, the DHCP Relay Agent forwards the request to the list of DHCP servers specified when the DHCP Relay Agent is started. When a DHCP server returns a reply, the reply is broadcast or unicast on the network that sent the original request.

The DHCP Relay Agent listens for DHCP requests on all interfaces unless the interfaces are specified in **/etc/sysconfig/dhcrelay** with the **INTERFACES** directive.

To start the DHCP Relay Agent, use the command **service dhcrelay start**.

16.3. CONFIGURING A DHCpv4 CLIENT

To configure a DHCP client manually, modify the **/etc/sysconfig/network** file to enable networking and the configuration file for each network device in the **/etc/sysconfig/network-scripts** directory. In this directory, each device should have a configuration file named **ifcfg-eth0**, where **eth0** is the network device name.

Make sure that the **/etc/sysconfig/network-scripts/ifcfg-eth0** file contains the following lines:

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
```

To use DHCP, set a configuration file for each device.

Other options for the network script include:

- **DHCP_HOSTNAME** – Only use this option if the DHCP server requires the client to specify a host name before receiving an IP address.
- **PEERDNS=<answer>**, where **<answer>** is one of the following:
 - **yes** – Modify **/etc/resolv.conf** with information from the server. This is the default.
 - **no** – Do not modify **/etc/resolv.conf**.

If you prefer using a graphical interface, see [Chapter 10, NetworkManager](#) for instructions on using **NetworkManager** to configure a network interface to use DHCP.



NOTE

For advanced configurations of client DHCP options such as protocol timing, lease requirements and requests, dynamic DNS support, aliases, as well as a wide variety of values to override, prepend, or append to client-side configurations, see the **dhclient** and **dhclient.conf** man pages.

16.4. CONFIGURING A MULTIHOME DHCP SERVER

A multihome DHCP server serves multiple networks, that is, multiple subnets. The examples in these sections detail how to configure a DHCP server to serve multiple networks, select which network interfaces to listen on, and how to define network settings for systems that move networks.

Before making any changes, back up the existing **/etc/sysconfig/dhcpd** and **/etc/dhcp/dhcpd.conf** files.

The DHCP daemon listens on all network interfaces unless otherwise specified. Use the **/etc/sysconfig/dhcpd** file to specify which network interfaces the DHCP daemon listens on. The following **/etc/sysconfig/dhcpd** example specifies that the DHCP daemon listens on the **eth0** and **eth1** interfaces:

```
DHCPDARGS="eth0 eth1";
```

If a system has three network interfaces cards – **eth0**, **eth1**, and **eth2** – and it is only desired that the DHCP daemon listens on the **eth0** card, then only specify **eth0** in **/etc/sysconfig/dhcpd**:

```
DHCPDARGS="eth0";
```

The following is a basic **/etc/dhcp/dhcpd.conf** file, for a server that has two network interfaces, **eth0** in a 10.0.0.0/24 network, and **eth1** in a 172.16.0.0/24 network. Multiple **subnet** declarations allow you to define different settings for multiple networks:

```
default-lease-time 600;
max-lease-time 7200;
subnet 10.0.0.0 netmask 255.255.255.0 {
    option subnet-mask 255.255.255.0;
    option routers 10.0.0.1;
    range 10.0.0.5 10.0.0.15;
}
subnet 172.16.0.0 netmask 255.255.255.0 {
    option subnet-mask 255.255.255.0;
    option routers 172.16.0.1;
    range 172.16.0.5 172.16.0.15;
}
```

subnet 10.0.0.0 netmask 255.255.255.0;

A **subnet** declaration is required for every network your DHCP server is serving. Multiple subnets require multiple **subnet** declarations. If the DHCP server does not have a network interface in a range of a **subnet** declaration, the DHCP server does not serve that network.

If there is only one **subnet** declaration, and no network interfaces are in the range of that subnet, the DHCP daemon fails to start, and an error such as the following is logged to **/var/log/messages**:

```
dhcpd: No subnet declaration for eth0 (0.0.0.0).
dhcpd: ** Ignoring requests on eth0.  If this is not what
dhcpd:   you want, please write a subnet declaration
dhcpd:   in your dhcpd.conf file for the network segment
dhcpd:   to which interface eth1 is attached. **
dhcpd:
dhcpd:
dhcpd: Not configured to listen on any interfaces!
```

option subnet-mask 255.255.255.0;

The **option subnet-mask** option defines a subnet mask, and overrides the **netmask** value in the **subnet** declaration. In simple cases, the subnet and netmask values are the same.

option routers 10.0.0.1;

The **option routers** option defines the default gateway for the subnet. This is required for systems to reach internal networks on a different subnet, as well as external networks.

range 10.0.0.5 10.0.0.15;

The **range** option specifies the pool of available IP addresses. Systems are assigned an address from the range of specified IP addresses.

For further information, see the **[dhcpd.conf\(5\)](#)** man page.

16.4.1. Host Configuration

Before making any changes, back up the existing **/etc/sysconfig/dhcpd** and **/etc/dhcp/dhcpd.conf** files.

Configuring a Single System for Multiple Networks

The following **/etc/dhcp/dhcpd.conf** example creates two subnets, and configures an IP address for the same system, depending on which network it connects to:

```
default-lease-time 600;
max-lease-time 7200;
subnet 10.0.0.0 netmask 255.255.255.0 {
    option subnet-mask 255.255.255.0;
    option routers 10.0.0.1;
    range 10.0.0.5 10.0.0.15;
}
subnet 172.16.0.0 netmask 255.255.255.0 {
    option subnet-mask 255.255.255.0;
    option routers 172.16.0.1;
    range 172.16.0.5 172.16.0.15;
}
host example0 {
    hardware ethernet 00:1A:6B:6A:2E:0B;
    fixed-address 10.0.0.20;
}
host example1 {
    hardware ethernet 00:1A:6B:6A:2E:0B;
    fixed-address 172.16.0.20;
}
```

host *example0*

The **host** declaration defines specific parameters for a single system, such as an IP address. To configure specific parameters for multiple hosts, use multiple **host** declarations.

Most DHCP clients ignore the name in **host** declarations, and as such, this name can be anything, as long as it is unique to other **host** declarations. To configure the same system for multiple networks, use a different name for each **host** declaration, otherwise the DHCP daemon fails to start. Systems are identified by the **hardware ethernet** option, not the name in the **host** declaration.

hardware ethernet 00:1A:6B:6A:2E:0B;

The **hardware ethernet** option identifies the system. To find this address, run the **ip link** command.

fixed-address 10.0.0.20;

The **fixed-address** option assigns a valid IP address to the system specified by the **hardware ethernet** option. This address must be outside the IP address pool specified with the **range** option.

If **option** statements do not end with a semicolon, the DHCP daemon fails to start, and an error such as the following is logged to **/var/log/messages**:

```
/etc/dhcp/dhcpd.conf line 20: semicolon expected.
dhcpd: }
dhcpd: ^
dhcpd: /etc/dhcp/dhcpd.conf line 38: unexpected end of file
dhcpd:
dhcpd: ^
dhcpd: Configuration file errors encountered -- exiting
```

Configuring Systems with Multiple Network Interfaces

The following **host** declarations configure a single system, which has multiple network interfaces, so that each interface receives the same IP address. This configuration will not work if both network interfaces are connected to the same network at the same time:

```
host interface0 {
    hardware ethernet 00:1a:6b:6a:2e:0b;
    fixed-address 10.0.0.18;
}
host interface1 {
    hardware ethernet 00:1A:6B:6A:27:3A;
    fixed-address 10.0.0.18;
}
```

For this example, **interface0** is the first network interface, and **interface1** is the second interface. The different **hardware ethernet** options identify each interface.

If such a system connects to another network, add more **host** declarations, remembering to:

- assign a valid **fixed-address** for the network the host is connecting to.
- make the name in the **host** declaration unique.

When a name given in a **host** declaration is not unique, the DHCP daemon fails to start, and an error such as the following is logged to **/var/log/messages**:

```
dhcpd: /etc/dhcp/dhcpd.conf line 31: host interface0: already exists
dhcpd: }
dhcpd: ^
dhcpd: Configuration file errors encountered -- exiting
```

This error was caused by having multiple **host interface0** declarations defined in **/etc/dhcp/dhcpd.conf**.

16.5. DHCP FOR IPV6 (DHCPV6)

The ISC DHCP includes support for IPv6 (DHCPv6) since the 4.x release with a DHCPv6 server, client and relay agent functionality. The server, client and relay agents support both IPv4 and IPv6. However, the client and the server can only manage one protocol at a time – for dual support they must be started

separately for IPv4 and IPv6.

16.5.1. Configuring a DHCPv6 Server

The DHCPv6 server configuration file is installed together with the `dhcp` package and it can be found at `/etc/dhcp/dhcpd6.conf`.

The sample server configuration file can be found at `/usr/share/doc/dhcp-<version>/dhcpd6.conf.sample`, in Red Hat Enterprise Linux 6 at `/usr/share/doc/dhcp-4.1.1/dhcpd6.conf.sample`.

A simple DHCPv6 server configuration file can look like this:

```
subnet6 2001:db8:0:1::/64 {
    range6 2001:db8:0:1::129 2001:db8:0:1::254;
    option dhcp6.name-servers fec0:0:0:1::1;
    option dhcp6.domain-search "domain.example";
}
```

For more examples, see the **`dhcpd.conf(5)`** man page.

To start the DHCPv6 service, enter the command **`service dhcpcd6 start`** as root. To stop the DHCPv6 server, use the command **`service dhcpcdv6 stop`**.

To pass command-line options to **`dhcpd`** daemon when the DHCPv6 service starts, use the `/etc/sysconfig/dhcpcd6` file. This file uses the same structure like the `/etc/sysconfig/dhcpd`:

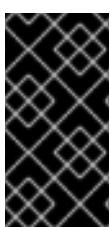
```
# cat /etc/sysconfig/dhcpcd6
# Command line options here
DHCPDARGS=
```

The value added to the **DHCPDARGS** option is passed to the DHCPv6 service, which passes it to the **`dhcpd`** daemon. For more information, see the **STANDARD DHCPV6 OPTIONS** section in the **`dhcpcd-options(5)`** man page. For additional examples, see the [Dynamic IPv6 configuration](#) on the Fedora Project wiki.

16.5.2. Configuring a DHCPv6 Client

The default configuration of the DHCPv6 client works fine in the most cases. However, to configure a DHCP client manually, create and modify the `/etc/dhcp/dhclient.conf` file. See the `/usr/share/doc/dhclient-4.1.1/dhclient6.conf.sample` for a client configuration file example.

For advanced configurations of DHCPv6 client options such as protocol timing, lease requirements and requests, dynamic DNS support, aliases, as well as a wide variety of values to override, prepend, or append to client-side configurations, see the **`dhclient.conf(5)`** man page and the **STANDARD DHCPV6 OPTIONS** section in the **`dhcpcd-options(5)`** man page.



IMPORTANT

In Red Hat Enterprise Linux 6, a DHCPv6 client is correctly handled only by **NetworkManager** and should not generally be run separately. That is because DHCPv6, unlike DHCPv4, is not a standalone network configuration protocol but is always supposed to be used together with router discovery.

16.6. ADDITIONAL RESOURCES

For additional information, see *The DHCP Handbook; Ralph Droms and Ted Lemon; 2003* or the following resources.

16.6.1. Installed Documentation

- **dhcpd** man page – Describes how the DHCP daemon works.
- **dhcpd.conf** man page – Explains how to configure the DHCP configuration file; includes some examples.
- **dhcpd.leases** man page – Describes a persistent database of leases.
- **dhcp-options** man page – Explains the syntax for declaring DHCP options in **dhcpd.conf**; includes some examples.
- **dhcrelay** man page – Explains the DHCP Relay Agent and its configuration options.
- **/usr/share/doc/dhcp-<version>/** – Contains sample files, README files, and release notes for current versions of the DHCP service.

CHAPTER 17. DNS SERVERS

DNS (Domain Name System), also known as a *nameserver*, is a network system that associates host names with their respective IP addresses. For users, this has the advantage that they can refer to machines on the network by names that are usually easier to remember than the numerical network addresses. For system administrators, using the nameserver allows them to change the IP address for a host without ever affecting the name-based queries, or to decide which machines handle these queries.

17.1. INTRODUCTION TO DNS

DNS is usually implemented using one or more centralized servers that are authoritative for certain domains. When a client host requests information from a nameserver, it usually connects to port 53. The nameserver then attempts to resolve the name requested. If it does not have an authoritative answer, or does not already have the answer cached from an earlier query, it queries other nameservers, called *root nameservers*, to determine which nameservers are authoritative for the name in question, and then queries them to get the requested name.

17.1.1. Nameserver Zones

In a DNS server such as BIND (Berkeley Internet Name Domain), all information is stored in basic data elements called *resource records* (RR). The resource record is usually a *fully qualified domain name* (FQDN) of a host, and is broken down into multiple sections organized into a tree-like hierarchy. This hierarchy consists of a main trunk, primary branches, secondary branches, and so on.

Example 17.1. A simple resource record

bob.sales.example.com

Each level of the hierarchy is divided by a period (that is, `.`). In [Example 17.1, “A simple resource record”](#), **com** defines the *top-level domain*, **example** its subdomain, and **sales** the subdomain of **example**. In this case, **bob** identifies a resource record that is part of the **sales.example.com** domain. With the exception of the part furthest to the left (that is, **bob**), each of these sections is called a *zone* and defines a specific *namespace*.

Zones are defined on authoritative nameservers through the use of *zone files*, which contain definitions of the resource records in each zone. Zone files are stored on *primary nameservers* (also called *master nameservers*), where changes are made to the files, and *secondary nameservers* (also called *slave nameservers*), which receive zone definitions from the primary nameservers. Both primary and secondary nameservers are authoritative for the zone and look the same to clients. Depending on the configuration, any nameserver can also serve as a primary or secondary server for multiple zones at the same time.

17.1.2. Nameserver Types

There are two nameserver configuration types:

authoritative

Authoritative nameservers answer to resource records that are part of their zones only. This category includes both primary (master) and secondary (slave) nameservers.

recursive

Recursive nameservers offer resolution services, but they are not authoritative for any zone. Answers for all resolutions are cached in memory for a fixed period of time, which is specified by the retrieved resource record.

Although a nameserver can be both authoritative and recursive at the same time, it is recommended not to combine the configuration types. To be able to perform their work, authoritative servers should be available to all clients all the time. On the other hand, since the recursive lookup takes far more time than authoritative responses, recursive servers should be available to a restricted number of clients only, otherwise they are prone to distributed denial of service (DDoS) attacks.

17.1.3. BIND as a Nameserver

BIND consists of a set of DNS-related programs. It contains a nameserver called **named**, an administration utility called **rndc**, and a debugging tool called **dig**. See [Chapter 12, Services and Daemons](#) for more information on how to run a service in Red Hat Enterprise Linux.

17.2. BIND

This chapter covers **BIND** (Berkeley Internet Name Domain), the DNS server included in Red Hat Enterprise Linux. It focuses on the structure of its configuration files, and describes how to administer it both locally and remotely.

17.2.1. Configuring the named Service

When the **named** service is started, it reads the configuration from the files as described in [Table 17.1, “The named service configuration files”](#).

Table 17.1. The named service configuration files

Path	Description
/etc/named.conf	The main configuration file.
/etc/named/	An auxiliary directory for configuration files that are included in the main configuration file.

The configuration file consists of a collection of statements with nested options surrounded by opening and closing curly brackets. Note that when editing the file, you have to be careful not to make any syntax error, otherwise the **named** service will not start. A typical **/etc/named.conf** file is organized as follows:

```

statement-1 ["statement-1-name"] [statement-1-class] {
    option-1;
    option-2;
    option-N;
};

statement-2 ["statement-2-name"] [statement-2-class] {
    option-1;
    option-2;
    option-N;
};

statement-N ["statement-N-name"] [statement-N-class] {
    option-1;
}

```

```

option-2;
option-N;
};

```

NOTE

If you have installed the bind-chroot package, the BIND service will run in the **/var/named/chroot** environment. In that case, the initialization script will mount the above configuration files using the **mount --bind** command, so that you can manage the configuration outside this environment. There is no need to copy anything into the **/var/named/chroot** directory because it is mounted automatically. This simplifies maintenance since you do not need to take any special care of **BIND** configuration files if it is run in a **chroot** environment. You can organize everything as you would with **BIND** not running in a **chroot** environment.

The following directories are automatically mounted into **/var/named/chroot** if they are empty in the **/var/named/chroot** directory. They must be kept empty if you want them to be mounted into **/var/named/chroot**:

- **/var/named**
- **/etc/pki/dnssec-keys**
- **/etc/named**
- **/usr/lib64/bind** or **/usr/lib/bind** (architecture dependent).

The following files are also mounted if the target file does not exist in **/var/named/chroot**.

- **/etc/named.conf**
- **/etc/rndc.conf**
- **/etc/rndc.key**
- **/etc/named.rfc1912.zones**
- **/etc/named.dnssec.keys**
- **/etc/named.iscdlv.key**
- **/etc/named.root.key**

17.2.1.1. Common Statement Types

The following types of statements are commonly used in **/etc/named.conf**:

acl

The **acl** (Access Control List) statement allows you to define groups of hosts, so that they can be permitted or denied access to the nameserver. It takes the following form:

```

acl acl-name {
    match-element;
    ...
};

```

The *acl-name* statement name is the name of the access control list, and the *match-element* option is usually an individual IP address (such as **10.0.1.1**) or a CIDR (Classless Inter-Domain Routing) network notation (for example, **10.0.1.0/24**). For a list of already defined keywords, see [Table 17.2, "Predefined access control lists"](#).

Table 17.2. Predefined access control lists

Keyword	Description
any	Matches every IP address.
localhost	Matches any IP address that is in use by the local system.
localnets	Matches any IP address on any network to which the local system is connected.
none	Does not match any IP address.

The **acl** statement can be especially useful in conjunction with other statements such as **options**. [Example 17.2, "Using acl in conjunction with options"](#) defines two access control lists, **black-hats** and **red-hats**, and adds **black-hats** on the blacklist while granting **red-hats** a normal access.

Example 17.2. Using acl in conjunction with options

```
acl black-hats {
    10.0.2.0/24;
    192.168.0.0/24;
    1234:5678::9abc/24;
};

acl red-hats {
    10.0.1.0/24;
};

options {
    blackhole { black-hats; };
    allow-query { red-hats; };
    allow-query-cache { red-hats; };
};
```

include

The **include** statement allows you to include files in the **/etc/named.conf**, so that potentially sensitive data can be placed in a separate file with restricted permissions. It takes the following form:

```
include "file-name";
```

The *file-name* statement name is an absolute path to a file.

Example 17.3. Including a file to /etc/named.conf

```
include "/etc/named.rfc1912.zones";
```

options

The **options** statement allows you to define global server configuration options as well as to set defaults for other statements. It can be used to specify the location of the **named** working directory, the types of queries allowed, and much more. It takes the following form:

```
options {
    option;
    ...
};
```

For a list of frequently used *option* directives, see [Table 17.3, “Commonly used options”](#) below.

Table 17.3. Commonly used options

Option	Description
allow-query	Specifies which hosts are allowed to query the nameserver for authoritative resource records. It accepts an access control list, a collection of IP addresses, or networks in the CIDR notation. All hosts are allowed by default.
allow-query-cache	Specifies which hosts are allowed to query the nameserver for non-authoritative data such as recursive queries. Only localhost and localnets are allowed by default.
blackhole	Specifies which hosts are <i>not</i> allowed to query the nameserver. This option should be used when particular host or network floods the server with requests. The default option is none .
directory	Specifies a working directory for the named service. The default option is /var/named/ .
dnssec-enable	Specifies whether to return DNSSEC related resource records. The default option is yes .
dnssec-validation	Specifies whether to prove that resource records are authentic via DNSSEC. The default option is yes .
forwarders	Specifies a list of valid IP addresses for nameservers to which the requests should be forwarded for resolution.
forward	Specifies the behavior of the forwarders directive. It accepts the following options: <ul style="list-style-type: none"> ● first – The server will query the nameservers listed in the forwarders directive before attempting to resolve the name on its own. ● only – When unable to query the nameservers listed in the forwarders directive, the server will not attempt to resolve the name on its own.

Option	Description
listen-on	Specifies the IPv4 network interface on which to listen for queries. On a DNS server that also acts as a gateway, you can use this option to answer queries originating from a single network only. All IPv4 interfaces are used by default.
listen-on-v6	Specifies the IPv6 network interface on which to listen for queries. On a DNS server that also acts as a gateway, you can use this option to answer queries originating from a single network only. All IPv6 interfaces are used by default.
max-cache-size	Specifies the maximum amount of memory to be used for server caches. When the limit is reached, the server causes records to expire prematurely so that the limit is not exceeded. In a server with multiple views, the limit applies separately to the cache of each view. The default option is 32M .
notify	Specifies whether to notify the secondary nameservers when a zone is updated. It accepts the following options: <ul style="list-style-type: none"> ● yes – The server will notify all secondary nameservers. ● no – The server will <i>not</i> notify any secondary nameserver. ● master-only – The server will notify primary server for the zone only. ● explicit – The server will notify only the secondary servers that are specified in the also-notify list within a zone statement.
pid-file	Specifies the location of the process ID file created by the named service.
recursion	Specifies whether to act as a recursive server. The default option is yes .
statistics-file	Specifies an alternate location for statistics files. The /var/named/named.stats file is used by default.



IMPORTANT

To prevent distributed denial of service (DDoS) attacks, it is recommended that you use the **allow-query-cache** option to restrict recursive DNS services for a particular subset of clients only.

See the *BIND 9 Administrator Reference Manual* referenced in [Section 17.2.7.1, “Installed Documentation”](#), and the **named.conf** manual page for a complete list of available options.

Example 17.4. Using the options statement

```
options {
    allow-query     { localhost; };
    listen-on port 53 { 127.0.0.1; };
    listen-on-v6 port 53 { ::1; };
    max-cache-size 256M;
    directory      "/var/named";
    statistics-file "/var/named/data/named_stats.txt";
```

```

recursion      yes;
dnssec-enable  yes;
dnssec-validation yes;
};

```

zone

The **zone** statement allows you to define the characteristics of a zone, such as the location of its configuration file and zone-specific options, and can be used to override the global **options** statements. It takes the following form:

```

zone zone-name [zone-class] {
    option;
    ...
};

```

The *zone-name* attribute is the name of the zone, *zone-class* is the optional class of the zone, and *option* is a **zone** statement option as described in [Table 17.4, “Commonly used options”](#).

The *zone-name* attribute is particularly important, as it is the default value assigned for the **\$ORIGIN** directive used within the corresponding zone file located in the **/var/named** directory. The **named** daemon appends the name of the zone to any non-fully qualified domain name listed in the zone file. For example, if a **zone** statement defines the namespace for **example.com**, use **example.com** as the *zone-name* so that it is placed at the end of host names within the **example.com** zone file.

For more information about zone files, see [Section 17.2.2, “Editing Zone Files”](#).

Table 17.4. Commonly used options

Option	Description
allow-query	Specifies which clients are allowed to request information about this zone. This option overrides global allow-query option. All query requests are allowed by default.
allow-transfer	Specifies which secondary servers are allowed to request a transfer of the zone's information. All transfer requests are allowed by default.
allow-update	Specifies which hosts are allowed to dynamically update information in their zone. The default option is to deny all dynamic update requests. Note that you should be careful when allowing hosts to update information about their zone. Do not set IP addresses in this option unless the server is in the trusted network. Instead, use TSIG key as described in Section 17.2.5.3, “Transaction SIGnatures (TSIG)” .
file	Specifies the name of the file in the named working directory that contains the zone's configuration data.
masters	Specifies from which IP addresses to request authoritative zone information. This option is used only if the zone is defined as type slave .

Option	Description
notify	<p>Specifies whether to notify the secondary nameservers when a zone is updated. It accepts the following options:</p> <ul style="list-style-type: none"> ● yes – The server will notify all secondary nameservers. ● no – The server will <i>not</i> notify any secondary nameserver. ● master-only – The server will notify primary server for the zone only. ● explicit – The server will notify only the secondary servers that are specified in the also-notify list within a zone statement.
type	<p>Specifies the zone type. It accepts the following options:</p> <ul style="list-style-type: none"> ● delegation-only – Enforces the delegation status of infrastructure zones such as COM, NET, or ORG. Any answer that is received without an explicit or implicit delegation is treated as NXDOMAIN. This option is only applicable in TLDs (Top-Level Domain) or root zone files used in recursive or caching implementations. ● forward – Forwards all requests for information about this zone to other nameservers. ● hint – A special type of zone used to point to the root nameservers which resolve queries when a zone is not otherwise known. No configuration beyond the default is necessary with a hint zone. ● master – Designates the nameserver as authoritative for this zone. A zone should be set as the master if the zone's configuration files reside on the system. ● slave – Designates the nameserver as a slave server for this zone. Master server is specified in masters directive.

Most changes to the **/etc/named.conf** file of a primary or secondary nameserver involve adding, modifying, or deleting **zone** statements, and only a small subset of **zone** statement options is usually needed for a nameserver to work efficiently.

In [Example 17.5, “A zone statement for a primary nameserver”](#), the zone is identified as **example.com**, the type is set to **master**, and the **named** service is instructed to read the **/var/named/example.com.zone** file. It also allows only a secondary nameserver (**192.168.0.2**) to transfer the zone.

Example 17.5. A zone statement for a primary nameserver

```
zone "example.com" IN {
    type master;
    file "example.com.zone";
    allow-transfer { 192.168.0.2; };
};
```

A secondary server's **zone** statement is slightly different. The type is set to **slave**, and the **masters** directive is telling **named** the IP address of the master server.

In [Example 17.6, "A zone statement for a secondary nameserver"](#), the **named** service is configured to query the primary server at the **192.168.0.1** IP address for information about the **example.com** zone. The received information is then saved to the **/var/named/slaves/example.com.zone** file. Note that you have to put all slave zones to **/var/named/slaves** directory, otherwise the service will fail to transfer the zone.

Example 17.6. A zone statement for a secondary nameserver

```
zone "example.com" {
    type slave;
    file "slaves/example.com.zone";
    masters { 192.168.0.1; };
};
```

17.2.1.2. Other Statement Types

The following types of statements are less commonly used in **/etc/named.conf**:

controls

The **controls** statement allows you to configure various security requirements necessary to use the **rndc** command to administer the **named** service.

See [Section 17.2.3, "Using the rndc Utility"](#) for more information on the **rndc** utility and its usage.

key

The **key** statement allows you to define a particular key by name. Keys are used to authenticate various actions, such as secure updates or the use of the **rndc** command. Two options are used with **key**:

- **algorithm algorithm-name** – The type of algorithm to be used (for example, **hmac-md5**).
- **secret "key-value"** – The encrypted key.

See [Section 17.2.3, "Using the rndc Utility"](#) for more information on the **rndc** utility and its usage.

logging

The **logging** statement allows you to use multiple types of logs, so called *channels*. By using the **channel** option within the statement, you can construct a customized type of log with its own file name (**file**), size limit (**size**), versioning (**version**), and level of importance (**severity**). Once a customized channel is defined, a **category** option is used to categorize the channel and begin logging when the **named** service is restarted.

By default, **named** sends standard messages to the **rsyslog** daemon, which places them in **/var/log/messages**. Several standard channels are built into BIND with various severity levels, such as **default_syslog** (which handles informational logging messages) and **default_debug** (which specifically handles debugging messages). A default category, called **default**, uses the built-in channels to do normal logging without any special configuration.

Customizing the logging process can be a very detailed process and is beyond the scope of this chapter. For information on creating custom BIND logs, see the *BIND 9 Administrator Reference Manual* referenced in [Section 17.2.7.1, “Installed Documentation”](#).

server

The **server** statement allows you to specify options that affect how the **named** service should respond to remote nameservers, especially with regard to notifications and zone transfers.

The **transfer-format** option controls the number of resource records that are sent with each message. It can be either **one-answer** (only one resource record), or **many-answers** (multiple resource records). Note that while the **many-answers** option is more efficient, it is not supported by older versions of BIND.

trusted-keys

The **trusted-keys** statement allows you to specify assorted public keys used for secure DNS (DNSSEC). See [Section 17.2.5.4, “DNS Security Extensions \(DNSSEC\)”](#) for more information on this topic.

view

The **view** statement allows you to create special views depending upon which network the host querying the nameserver is on. This allows some hosts to receive one answer regarding a zone while other hosts receive totally different information. Alternatively, certain zones may only be made available to particular trusted hosts while non-trusted hosts can only make queries for other zones.

Multiple views can be used as long as their names are unique. The **match-clients** option allows you to specify the IP addresses that apply to a particular view. If the **options** statement is used within a view, it overrides the already configured global options. Finally, most **view** statements contain multiple **zone** statements that apply to the **match-clients** list.

Note that the order in which the **view** statements are listed is important, as the first statement that matches a particular client's IP address is used. For more information on this topic, see [Section 17.2.5.1, “Multiple Views”](#).

17.2.1.3. Comment Tags

Additionally to statements, the **/etc/named.conf** file can also contain comments. Comments are ignored by the **named** service, but can prove useful when providing additional information to a user. The following are valid comment tags:

//

Any text after the // characters to the end of the line is considered a comment. For example:

 | notify yes; // notify all secondary nameservers

#

Any text after the # character to the end of the line is considered a comment. For example:

 | notify yes; # notify all secondary nameservers

/* and */

Any block of text enclosed in /* and */ is considered a comment. For example:

```
notify yes; /* notify all secondary nameservers */
```

17.2.2. Editing Zone Files

As outlined in [Section 17.1.1, “Nameserver Zones”](#), zone files contain information about a namespace. They are stored in the **named** working directory located in **/var/named**/ by default, and each zone file is named according to the **file** option in the **zone** statement, usually in a way that relates to the domain in question and identifies the file as containing zone data, such as **example.com.zone**.

Table 17.5. The named service zone files

Path	Description
/var/named	The working directory for the named service. The nameserver is <i>not</i> allowed to write to this directory.
/var/named/slaves	The directory for secondary zones. This directory is writable by the named service.
/var/named/dynamic	The directory for other files, such as dynamic DNS (DDNS) zones or managed DNSSEC keys. This directory is writable by the named service.
/var/named/data	The directory for various statistics and debugging files. This directory is writable by the named service.

A zone file consists of directives and resource records. Directives tell the nameserver to perform tasks or apply special settings to the zone, resource records define the parameters of the zone and assign identities to individual hosts. While the directives are optional, the resource records are required in order to provide name service to a zone.

All directives and resource records should be entered on individual lines.

17.2.2.1. Common Directives

Directives begin with the dollar sign character followed by the name of the directive, and usually appear at the top of the file. The following directives are commonly used in zone files:

\$INCLUDE

The **\$INCLUDE** directive allows you to include another file at the place where it appears, so that other zone settings can be stored in a separate zone file.

Example 17.7. Using the \$INCLUDE directive

```
$INCLUDE /var/named/penguin.example.com
```

\$ORIGIN

The **\$ORIGIN** directive allows you to append the domain name to unqualified records, such as those with the host name only. Note that the use of this directive is not necessary if the zone is specified in **/etc/named.conf**, since the zone name is used by default.

In [Example 17.8, “Using the \\$ORIGIN directive”](#), any names used in resource records that do not end in a trailing period are appended with **example.com**.

Example 17.8. Using the \$ORIGIN directive

```
$ORIGIN example.com.
```

\$TTL

The **\$TTL** directive allows you to set the default *Time to Live* (TTL) value for the zone, that is, how long is a zone record valid. Each resource record can contain its own TTL value, which overrides this directive.

Increasing this value allows remote nameservers to cache the zone information for a longer period of time, reducing the number of queries for the zone and lengthening the amount of time required to propagate resource record changes.

Example 17.9. Using the \$TTL directive

```
$TTL 1D
```

17.2.2.2. Common Resource Records

The following resource records are commonly used in zone files:

A

The *Address* record specifies an IP address to be assigned to a name. It takes the following form:

```
hostname IN A IP-address
```

If the *hostname* value is omitted, the record will point to the last specified *hostname*.

In [Example 17.10, “Using the A resource record”](#), the requests for **server1.example.com** are pointed to **10.0.1.3** or **10.0.1.5**.

Example 17.10. Using the A resource record

```
server1 IN A 10.0.1.3
          IN A 10.0.1.5
```

CNAME

The *Canonical Name* record maps one name to another. Because of this, this type of record is sometimes referred to as an *alias record*. It takes the following form:

alias-name IN CNAME real-name

CNAME records are most commonly used to point to services that use a common naming scheme, such as **www** for Web servers. However, there are multiple restrictions for their usage:

- CNAME records should not point to other CNAME records. This is mainly to avoid possible infinite loops.
- CNAME records should not contain other resource record types (such as A, NS, MX, etc.). The only exception are DNSSEC related records (that is, RRSIG, NSEC, etc.) when the zone is signed.
- Other resource record that point to the fully qualified domain name (FQDN) of a host (that is, NS, MX, PTR) should not point to a CNAME record.

In [Example 17.11, “Using the CNAME resource record”](#), the **A** record binds a host name to an IP address, while the **CNAME** record points the commonly used **www** host name to it.

Example 17.11. Using the CNAME resource record

```
server1 IN A 10.0.1.5
www    IN CNAME server1
```

MX

The *Mail Exchange* record specifies where the mail sent to a particular namespace controlled by this zone should go. It takes the following form:

IN MX preference-value email-server-name

The *email-server-name* is a fully qualified domain name (FQDN). The *preference-value* allows numerical ranking of the email servers for a namespace, giving preference to some email systems over others. The **MX** resource record with the lowest *preference-value* is preferred over the others. However, multiple email servers can possess the same value to distribute email traffic evenly among them.

In [Example 17.12, “Using the MX resource record”](#), the first **mail.example.com** email server is preferred to the **mail2.example.com** email server when receiving email destined for the **example.com** domain.

Example 17.12. Using the MX resource record

```
example.com. IN MX 10 mail.example.com.
                  IN MX 20 mail2.example.com.
```

NS

The *Nameserver* record announces authoritative nameservers for a particular zone. It takes the following form:

IN NS nameserver-name

The *nameserver-name* should be a fully qualified domain name (FQDN). Note that when two nameservers are listed as authoritative for the domain, it is not important whether these nameservers are secondary nameservers, or if one of them is a primary server. They are both still considered authoritative.

Example 17.13. Using the NS resource record

```
IN NS dns1.example.com.  
IN NS dns2.example.com.
```

PTR

The *Pointer* record points to another part of the namespace. It takes the following form:

```
last-IP-digit IN PTR FQDN-of-system
```

The *last-IP-digit* directive is the last number in an IP address, and the *FQDN-of-system* is a fully qualified domain name (FQDN).

PTR records are primarily used for reverse name resolution, as they point IP addresses back to a particular name. See [Section 17.2.2.4.2, “A Reverse Name Resolution Zone File”](#) for more examples of **PTR** records in use.

SOA

The *Start of Authority* record announces important authoritative information about a namespace to the nameserver. Located after the directives, it is the first resource record in a zone file. It takes the following form:

```
@ IN SOA primary-name-server hostmaster-email (  
    serial-number  
    time-to-refresh  
    time-to-retry  
    time-to-expire  
    minimum-TTL )
```

The directives are as follows:

- The @ symbol places the **\$ORIGIN** directive (or the zone's name if the **\$ORIGIN** directive is not set) as the namespace being defined by this **SOA** resource record.
- The *primary-name-server* directive is the host name of the primary nameserver that is authoritative for this domain.
- The *hostmaster-email* directive is the email of the person to contact about the namespace.
- The *serial-number* directive is a numerical value incremented every time the zone file is altered to indicate it is time for the **named** service to reload the zone.
- The *time-to-refresh* directive is the numerical value secondary nameservers use to determine how long to wait before asking the primary nameserver if any changes have been made to the zone.
- The *time-to-retry* directive is a numerical value used by secondary nameservers to determine

the length of time to wait before issuing a refresh request in the event that the primary nameserver is not answering. If the primary server has not replied to a refresh request before the amount of time specified in the *time-to-expire* directive elapses, the secondary servers stop responding as an authority for requests concerning that namespace.

- In BIND 4 and 8, the *minimum-TTL* directive is the amount of time other nameservers cache the zone's information. In BIND 9, it defines how long negative answers are cached for. Caching of negative answers can be set to a maximum of 3 hours (that is, **3H**).

When configuring BIND, all times are specified in seconds. However, it is possible to use abbreviations when specifying units of time other than seconds, such as minutes (**M**), hours (**H**), days (**D**), and weeks (**W**). [Table 17.6, "Seconds compared to other time units"](#) shows an amount of time in seconds and the equivalent time in another format.

Table 17.6. Seconds compared to other time units

Seconds	Other Time Units
60	1M
1800	30M
3600	1H
10800	3H
21600	6H
43200	12H
86400	1D
259200	3D
604800	1W
31536000	365D

Example 17.14. Using the SOA resource record

```
@ IN SOA dns1.example.com. hostmaster.example.com. (
    2001062501 ; serial
    21600      ; refresh after 6 hours
    3600       ; retry after 1 hour
    604800     ; expire after 1 week
    86400 )    ; minimum TTL of 1 day
```

17.2.2.3. Comment Tags

Additionally to resource records and directives, a zone file can also contain comments. Comments are ignored by the **named** service, but can prove useful when providing additional information to the user. Any text after the semicolon character to the end of the line is considered a comment. For example:

```
604800 ; expire after 1 week
```

17.2.2.4. Example Usage

The following examples show the basic usage of zone files.

17.2.2.4.1. A Simple Zone File

[Example 17.15, "A simple zone file"](#) demonstrates the use of standard directives and **SOA** values.

Example 17.15. A simple zone file

```
$ORIGIN example.com.
$TTL 86400
@ IN SOA dns1.example.com. hostmaster.example.com. (
    2001062501 ; serial
    21600      ; refresh after 6 hours
    3600       ; retry after 1 hour
    604800     ; expire after 1 week
    86400 )    ; minimum TTL of 1 day
;
;
    IN NS dns1.example.com.
    IN NS dns2.example.com.
dns1 IN A 10.0.1.1
    IN AAAA aaaa:bbbb::1
dns2 IN A 10.0.1.2
    IN AAAA aaaa:bbbb::2
;
;
@ IN MX 10 mail.example.com.
    IN MX 20 mail2.example.com.
mail IN A 10.0.1.5
    IN AAAA aaaa:bbbb::5
mail2 IN A 10.0.1.6
    IN AAAA aaaa:bbbb::6
;
;
; This sample zone file illustrates sharing the same IP addresses
; for multiple services:
;
services IN A 10.0.1.10
    IN AAAA aaaa:bbbb::10
    IN A 10.0.1.11
    IN AAAA aaaa:bbbb::11
;
;
ftp IN CNAME services.example.com.
www IN CNAME services.example.com.
;
;
```

In this example, the authoritative nameservers are set as **dns1.example.com** and **dns2.example.com**, and are tied to the **10.0.1.1** and **10.0.1.2** IP addresses respectively using the **A** record.

The email servers configured with the **MX** records point to **mail** and **mail2** via **A** records. Since these names do not end in a trailing period, the **\$ORIGIN** domain is placed after them, expanding them to **mail.example.com** and **mail2.example.com**.

Services available at the standard names, such as **www.example.com** (WWW), are pointed at the appropriate servers using the **CNAME** record.

This zone file would be called into service with a **zone** statement in the **/etc/named.conf** similar to the following:

```
zone "example.com" IN {
    type master;
    file "example.com.zone";
    allow-update { none; };
};
```

17.2.2.4.2. A Reverse Name Resolution Zone File

A reverse name resolution zone file is used to translate an IP address in a particular namespace into a fully qualified domain name (FQDN). It looks very similar to a standard zone file, except that the **PTR** resource records are used to link the IP addresses to a fully qualified domain name as shown in [Example 17.16, “A reverse name resolution zone file”](#).

Example 17.16. A reverse name resolution zone file

```
$ORIGIN 1.0.10.in-addr.arpa.
$TTL 86400
@ IN SOA dns1.example.com. hostmaster.example.com. (
    2001062501 ; serial
    21600      ; refresh after 6 hours
    3600       ; retry after 1 hour
    604800     ; expire after 1 week
    86400 )    ; minimum TTL of 1 day
;
@ IN NS dns1.example.com.
;
1 IN PTR dns1.example.com.
2 IN PTR dns2.example.com.
;
5 IN PTR server1.example.com.
6 IN PTR server2.example.com.
;
3 IN PTR ftp.example.com.
4 IN PTR ftp.example.com.
```

In this example, IP addresses **10.0.1.1** through **10.0.1.6** are pointed to the corresponding fully qualified domain name.

This zone file would be called into service with a **zone** statement in the **/etc/named.conf** file similar to the following:

```
zone "1.0.10.in-addr.arpa" IN {
    type master;
    file "example.com.rr.zone";
    allow-update { none; };
};
```

There is very little difference between this example and a standard **zone** statement, except for the zone name. Note that a reverse name resolution zone requires the first three blocks of the IP address reversed followed by **.in-addr.arpa**. This allows the single block of IP numbers used in the reverse name resolution zone file to be associated with the zone.

17.2.3. Using the **rndc** Utility

The **rndc** utility is a command-line tool that allows you to administer the **named** service, both locally and from a remote machine. Its usage is as follows:

```
rndc [option...] command [command-option]
```

17.2.3.1. Configuring the Utility

To prevent unauthorized access to the service, **named** must be configured to listen on the selected port (that is, **953** by default), and an identical key must be used by both the service and the **rndc** utility.

Table 17.7. Relevant files

Path	Description
/etc/named.conf	The default configuration file for the named service.
/etc/rndc.conf	The default configuration file for the rndc utility.
/etc/rndc.key	The default key location.

The **rndc** configuration is located in **/etc/rndc.conf**. If the file does not exist, the utility will use the key located in **/etc/rndc.key**, which was generated automatically during the installation process using the **rndc-confgen -a** command.

The **named** service is configured using the **controls** statement in the **/etc/named.conf** configuration file as described in [Section 17.2.1.2, “Other Statement Types”](#). Unless this statement is present, only the connections from the loopback address (that is, **127.0.0.1**) will be allowed, and the key located in **/etc/rndc.key** will be used.

For more information on this topic, see manual pages and the *BIND 9 Administrator Reference Manual* listed in [Section 17.2.7, “Additional Resources”](#).



IMPORTANT

To prevent unprivileged users from sending control commands to the service, make sure only root is allowed to read the **/etc/rndc.key** file:

```
~]# chmod o-rwx /etc/rndc.key
```

17.2.3.2. Checking the Service Status

To check the current status of the **named** service, use the following command:

```
~]# rndc status
version: 9.7.0-P2-RedHat-9.7.0-5.P2.el6
CPUs found: 1
worker threads: 1
number of zones: 16
debug level: 0
xfers running: 0
xfers deferred: 0
soa queries in progress: 0
query logging is OFF
recursive clients: 0/0/1000
tcp clients: 0/100
server is up and running
```

17.2.3.3. Reloading the Configuration and Zones

To reload both the configuration file and zones, type the following at a shell prompt:

```
~]# rndc reload
server reload successful
```

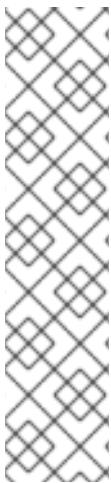
This will reload the zones while keeping all previously cached responses, so that you can make changes to the zone files without losing all stored name resolutions.

To reload a single zone, specify its name after the **reload** command, for example:

```
~]# rndc reload localhost
zone reload up-to-date
```

Finally, to reload the configuration file and newly added zones only, type:

```
~]# rndc reconfig
```



NOTE

If you intend to manually modify a zone that uses Dynamic DNS (DDNS), make sure you run the **freeze** command first:

```
~]# rndc freeze localhost
```

Once you are finished, run the **thaw** command to allow the DDNS again and reload the zone:

```
~]# rndc thaw localhost
```

The zone reload and thaw was successful.

17.2.3.4. Updating Zone Keys

To update the DNSSEC keys and sign the zone, use the **sign** command. For example:

```
~]# rndc sign localhost
```

Note that to sign a zone with the above command, the **auto-dnssec** option has to be set to **maintain** in the zone statement. For instance:

```
zone "localhost" IN {
    type master;
    file "named.localhost";
    allow-update { none; };
    auto-dnssec maintain;
};
```

17.2.3.5. Enabling the DNSSEC Validation

To enable the DNSSEC validation, type the following at a shell prompt:

```
~]# rndc validation on
```

Similarly, to disable this option, type:

```
~]# rndc validation off
```

See the **options** statement described in [Section 17.2.1.1, “Common Statement Types”](#) for information on how to configure this option in **/etc/named.conf**.

17.2.3.6. Enabling the Query Logging

To enable (or disable in case it is currently enabled) the query logging, run the following command:

```
~]# rndc querylog
```

To check the current setting, use the **status** command as described in [Section 17.2.3.2, “Checking the Service Status”](#).

17.2.4. Using the dig Utility

The **dig** utility is a command-line tool that allows you to perform DNS lookups and debug a nameserver configuration. Its typical usage is as follows:

```
dig [@server] [option...] name type
```

See [Section 17.2.2.2, “Common Resource Records”](#) for a list of common types.

17.2.4.1. Looking Up a Nameserver

To look up a nameserver for a particular domain, use the command in the following form:

```
dig name NS
```

In [Example 17.17, “A sample nameserver lookup”](#), the **dig** utility is used to display nameservers for **example.com**.

Example 17.17. A sample nameserver lookup

```
~]$ dig example.com NS
; <>>> DiG 9.7.1-P2-RedHat-9.7.1-2.P2.fc13 <>> example.com NS
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57883
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0
;;
;; QUESTION SECTION:
;example.com.           IN      NS
;;
;; ANSWER SECTION:
example.com.      99374  IN      NS      a.iana-servers.net.
example.com.      99374  IN      NS      b.iana-servers.net.
;;
;; Query time: 1 msec
;; SERVER: 10.34.255.7#53(10.34.255.7)
;; WHEN: Wed Aug 18 18:04:06 2010
;; MSG SIZE rcvd: 77
```

17.2.4.2. Looking Up an IP Address

To look up an IP address assigned to a particular domain, use the command in the following form:

```
dig name A
```

In [Example 17.18, “A sample IP address lookup”](#), the **dig** utility is used to display the IP address of **example.com**.

Example 17.18. A sample IP address lookup

```
~]$ dig example.com A
```

```
; <>> DiG 9.7.1-P2-RedHat-9.7.1-2.P2.fc13 <>> example.com A
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4849
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 0

;; QUESTION SECTION:
;example.com.           IN      A

;; ANSWER SECTION:
example.com.      155606 IN      A      192.0.32.10

;; AUTHORITY SECTION:
example.com.      99175  IN      NS      a.iana-servers.net.
example.com.      99175  IN      NS      b.iana-servers.net.

;; Query time: 1 msec
;; SERVER: 10.34.255.7#53(10.34.255.7)
;; WHEN: Wed Aug 18 18:07:25 2010
;; MSG SIZE rcvd: 93
```

17.2.4.3. Looking Up a Hostname

To look up a host name for a particular IP address, use the command in the following form:

```
dig -x address
```

In [Example 17.19, "A sample host name lookup"](#), the **dig** utility is used to display the host name assigned to **192.0.32.10**.

Example 17.19. A sample host name lookup

```
~]$ dig -x 192.0.32.10

; <>> DiG 9.7.1-P2-RedHat-9.7.1-2.P2.fc13 <>> -x 192.0.32.10
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29683
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 6

;; QUESTION SECTION:
;10.32.0.192.in-addr.arpa.    IN      PTR

;; ANSWER SECTION:
10.32.0.192.in-addr.arpa. 21600 IN      PTR      www.example.com.

;; AUTHORITY SECTION:
32.0.192.in-addr.arpa. 21600 IN      NS      b.iana-servers.org.
32.0.192.in-addr.arpa. 21600 IN      NS      c.iana-servers.net.
32.0.192.in-addr.arpa. 21600 IN      NS      d.iana-servers.net.
32.0.192.in-addr.arpa. 21600 IN      NS      ns.icann.org.
32.0.192.in-addr.arpa. 21600 IN      NS      a.iana-servers.net.
```

```

;; ADDITIONAL SECTION:
a.iana-servers.net. 13688 IN A 192.0.34.43
b.iana-servers.org. 5844 IN A 193.0.0.236
b.iana-servers.org. 5844 IN AAAA 2001:610:240:2::c100:ec
c.iana-servers.net. 12173 IN A 139.91.1.10
c.iana-servers.net. 12173 IN AAAA 2001:648:2c30::1:10
ns.icann.org. 12884 IN A 192.0.34.126

;; Query time: 156 msec
;; SERVER: 10.34.255.7#53(10.34.255.7)
;; WHEN: Wed Aug 18 18:25:15 2010
;; MSG SIZE rcvd: 310

```

17.2.5. Advanced Features of BIND

Most BIND implementations only use the **named** service to provide name resolution services or to act as an authority for a particular domain. However, BIND version 9 has a number of advanced features that allow for a more secure and efficient DNS service.



IMPORTANT

Before attempting to use advanced features like DNSSEC, TSIG, or IXFR (Incremental Zone Transfer), make sure that the particular feature is supported by all nameservers in the network environment, especially when you use older versions of BIND or non-BIND servers.

All of the features mentioned are discussed in greater detail in the *BIND 9 Administrator Reference Manual* referenced in [Section 17.2.7.1, “Installed Documentation”](#).

17.2.5.1. Multiple Views

Optionally, different information can be presented to a client depending on the network a request originates from. This is primarily used to deny sensitive DNS entries from clients outside of the local network, while allowing queries from clients inside the local network.

To configure multiple views, add the **view** statement to the **/etc/named.conf** configuration file. Use the **match-clients** option to match IP addresses or entire networks and give them special options and zone data.

17.2.5.2. Incremental Zone Transfers (IXFR)

Incremental Zone Transfers (IXFR) allow a secondary nameserver to only download the updated portions of a zone modified on a primary nameserver. Compared to the standard transfer process, this makes the notification and update process much more efficient.

Note that IXFR is only available when using dynamic updating to make changes to master zone records. If manually editing zone files to make changes, *Automatic Zone Transfer (AXFR)* is used.

17.2.5.3. Transaction SIGnatures (TSIG)

Transaction SIGnatures (TSIG) ensure that a shared secret key exists on both primary and secondary

nameserver before allowing a transfer. This strengthens the standard IP address-based method of transfer authorization, since attackers would not only need to have access to the IP address to transfer the zone, but they would also need to know the secret key.

Since version 9, BIND also supports *TKEY*, which is another shared secret key method of authorizing zone transfers.



IMPORTANT

When communicating over an insecure network, do not rely on IP address-based authentication only.

17.2.5.4. DNS Security Extensions (DNSSEC)

Domain Name System Security Extensions (DNSSEC) provide origin authentication of DNS data, authenticated denial of existence, and data integrity. When a particular domain is marked as secure, the **SERVFAIL** response is returned for each resource record that fails the validation.

Note that to debug a DNSSEC-signed domain or a DNSSEC-aware resolver, you can use the **dig** utility as described in [Section 17.2.4, “Using the dig Utility”](#). Useful options are **+dnssec** (requests DNSSEC-related resource records by setting the DNSSEC OK bit), **+cd** (tells recursive nameserver not to validate the response), and **+bufsize=512** (changes the packet size to 512B to get through some firewalls).

17.2.5.5. Internet Protocol version 6 (IPv6)

Internet Protocol version 6 (IPv6) is supported through the use of **AAAA** resource records, and the **listen-on-v6** directive as described in [Table 17.3, “Commonly used options”](#).

17.2.6. Common Mistakes to Avoid

The following is a list of recommendations on how to avoid common mistakes users make when configuring a nameserver:

Use semicolons and curly brackets correctly

An omitted semicolon or unmatched curly bracket in the **/etc/named.conf** file can prevent the **named** service from starting.

Use period correctly

In zone files, a period at the end of a domain name denotes a fully qualified domain name. If omitted, the **named** service will append the name of the zone or the value of **\$ORIGIN** to complete it.

Increment the serial number when editing a zone file

If the serial number is not incremented, the primary nameserver will have the correct, new information, but the secondary nameservers will never be notified of the change, and will not attempt to refresh their data of that zone.

Configure the firewall

If a firewall is blocking connections from the **named** service to other nameservers, the recommended practice is to change the firewall settings.



WARNING

According to the recent research in DNS security, using a fixed UDP source port for DNS queries is a potential security vulnerability that could allow an attacker to conduct cache-poisoning attacks more easily. To prevent this, configure your firewall to allow queries from a random UDP source port.

17.2.7. Additional Resources

The following sources of information provide additional resources regarding BIND.

17.2.7.1. Installed Documentation

BIND features a full range of installed documentation covering many different topics, each placed in its own subject directory. For each item below, replace *version* with the version of the bind package installed on the system:

/usr/share/doc/bind-*version*/

The main directory containing the most recent documentation.

/usr/share/doc/bind-*version*/arm/

The directory containing the *BIND 9 Administrator Reference Manual* in HTML and SGML formats, which details BIND resource requirements, how to configure different types of nameservers, how to perform load balancing, and other advanced topics. For most new users of BIND, this is the best place to start.

/usr/share/doc/bind-*version*/draft/

The directory containing assorted technical documents that review issues related to the DNS service, and propose some methods to address them.

/usr/share/doc/bind-*version*/misc/

The directory designed to address specific advanced issues. Users of BIND version 8 should consult the **migration** document for specific changes they must make when moving to BIND 9. The **options** file lists all of the options implemented in BIND 9 that are used in **/etc/named.conf**.

/usr/share/doc/bind-*version*/rfc/

The directory providing every RFC document related to BIND.

There is also a number of man pages for the various applications and configuration files involved with BIND:

man rndc

The manual page for **rndc** containing the full documentation on its usage.

man named

The manual page for **named** containing the documentation on assorted arguments that can be used to control the BIND nameserver daemon.

man lwresd

The manual page for **lwresd** containing the full documentation on the lightweight resolver daemon and its usage.

man named.conf

The manual page with a comprehensive list of options available within the **named** configuration file.

man rndc.conf

The manual page with a comprehensive list of options available within the **rndc** configuration file.

17.2.7.2. Useful Websites

<http://www.isc.org/software/bind>

The home page of the BIND project containing information about current releases as well as a PDF version of the *BIND 9 Administrator Reference Manual*.

17.2.7.3. Related Books

DNS and BIND by Paul Albitz and Cricket Liu; O'Reilly & Associates

A popular reference that explains both common and esoteric BIND configuration options, and provides strategies for securing a DNS server.

The Concise Guide to DNS and BIND by Nicolai Langfeldt; Que

Looks at the connection between multiple network services and BIND, with an emphasis on task-oriented, technical topics.

CHAPTER 18. WEB SERVERS

HTTP (Hypertext Transfer Protocol) server, or a *web server*, is a network service that serves content to a client over the web. This typically means web pages, but any other documents can be served as well.

The web servers available in Red Hat Enterprise Linux 6 are:

- Apache HTTP Server
- nginx



IMPORTANT

Note that the nginx web server is available only as a Software Collection for Red Hat Enterprise Linux 6. See the [Red Hat Software Collections](#) Release Notes for information regarding getting access to nginx, usage of Software Collections, and other.

18.1. THE APACHE HTTP SERVER

This section focuses on the **Apache HTTP Server 2.2**, a robust, full-featured open source web server developed by the [Apache Software Foundation](#), that is included in Red Hat Enterprise Linux 6. It describes the basic configuration of the **httpd** service, and covers 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 2.2 and version 2.0, and if you are upgrading from a previous release of Red Hat Enterprise Linux, you will need to update the **httpd** service configuration accordingly. This section reviews some of the newly added features, outlines important changes, and guides you through the update of older configuration files.

18.1.1. New Features

The Apache HTTP Server version 2.2 introduces the following enhancements:

- Improved caching modules, that is, **mod_cache** and **mod_disk_cache**.
- Support for proxy load balancing, that is, the **mod_proxy_balancer** module.
- Support for large files on 32-bit architectures, allowing the web server to handle files greater than 2GB.
- A new structure for authentication and authorization support, replacing the authentication modules provided in previous versions.

18.1.2. Notable Changes

Since version 2.0, few changes have been made to the default **httpd** service configuration:

- The following modules are no longer loaded by default: **mod_cern_meta** and **mod_asis**.
- The following module is newly loaded by default: **mod_ext_filter**.

18.1.3. Updating the Configuration

To update the configuration files from the Apache HTTP Server version 2.0, 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, see [Section 18.1.9, “Enabling the mod_ssl Module”](#) for important information on enabling the Secure Sockets Layer (SSL) protocol.

Note that you can check the configuration for possible errors by using the following command:

```
~]# service httpd configtest  
Syntax OK
```

For more information on upgrading the Apache HTTP Server configuration from version 2.0 to 2.2, see <http://httpd.apache.org/docs/2.2/upgrading.html>.

18.1.4. Running the **httpd** Service

This section describes how to start, stop, restart, and check the current status of the Apache HTTP Server. To be able to use the **httpd** service, make sure you have the **httpd** installed. You can do so by using the following command:

```
~]# yum install httpd
```

For more information on the concept of runlevels and how to manage system services in Red Hat Enterprise Linux in general, see [Chapter 12, Services and Daemons](#).

18.1.4.1. Starting the Service

To run the **httpd** service, type the following at a shell prompt as **root**:

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

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

```
~]# chkconfig httpd on
```

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



NOTE

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.

18.1.4.2. Stopping the Service

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

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

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

```
~]# chkconfig httpd off
```

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

18.1.4.3. 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**:

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

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:

```
~]# service httpd reload
```

This causes the running **httpd** service to reload its configuration file. Any requests being currently processed will be interrupted, which may cause a client browser to display an error message or render a partial page.

3. To reload the configuration without affecting active requests, enter the following command as **root**:

```
~]# service httpd graceful
```

This causes the running **httpd** service to reload its configuration file. Any requests being currently processed will use the old configuration.

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

18.1.4.4. Verifying the Service Status

To verify that the **httpd** service is running, type the following at a shell prompt:

```
~]# service httpd status  
httpd (pid 19014) is running...
```

Alternatively, you can use the **Service Configuration** utility as described in [Section 12.2.1, “Using the Service Configuration Utility”](#).

18.1.5. Editing the Configuration Files

When the **httpd** service is started, by default, it reads the configuration from locations that are listed in [Table 18.1, “The httpd service configuration files”](#).

Table 18.1. The httpd service configuration files

Path	Description
/etc/httpd/conf/httpd.conf	The main configuration file.
/etc/httpd/conf.d/	An auxiliary directory for configuration files that are included in the main configuration file.

Although the default configuration should be suitable for most situations, it is a good idea to become at least familiar with some of the more important configuration options. Note that for any changes to take effect, the web server has to be restarted first. See [Section 18.1.4.3, “Restarting the Service”](#) for more information on how to restart the **httpd** service.

To check the configuration for possible errors, type the following at a shell prompt:

```
~]# service httpd configtest
Syntax OK
```

To make the recovery from mistakes easier, it is recommended that you make a copy of the original file before editing it.

18.1.5.1. 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 directorydirective
  ...
</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 18.1. 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 18.2. 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 18.3. 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 18.4. 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 18.5. 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 18.2, “Available <VirtualHost> options”](#).

Table 18.2. Available <VirtualHost> options

Option	Description
*	Represents all IP addresses.
default	Represents unmatched IP addresses.

Example 18.6. 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 18.7. 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 18.8. 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 18.9. 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 18.10. 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 a 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 18.11. 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 18.3. Available AddIcon options

Option	Description
^^DIRECTORY^^	Represents a directory.
^^BLANKICON^^	Represents a blank line.

Example 18.12. 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 18.13. 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 18.14. 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 18.15. 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 18.16. 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 18.17. 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 18.18. 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 18.4, "Available AllowOverride options"](#).

Table 18.4. 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 .

Option	Description
FileInfo	Allows the use of file type, metadata, and mod_rewrite 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 [=option,...]	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 18.19. 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 18.20. 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 18.21. 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 18.22. 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 18.5, "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 18.5. Available cache types

Type	Description
mem	The memory-based storage manager.
disk	The disk-based storage manager.
fd	The file descriptor cache.

Example 18.23. 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 18.24. 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 18.25. 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 18.6, "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 18.6. Available CacheNegotiatedDocs options

Option	Description
On	Enables caching the content-negotiated documents.
Off	Disables caching the content-negotiated documents.

Example 18.26. 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 18.27. 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 18.28. 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 18.29. 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 18.30. 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 18.31. 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 18.32. 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 18.33. 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 18.34. 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 18.35. 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 18.7, “Available ExtendedStatus options”](#). The default option is **Off**.

Table 18.7. Available ExtendedStatus options

Option	Description
On	Enables generating the detailed server status.
Off	Disables generating the detailed server status.

Example 18.36. 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 18.37. 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 18.38. 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 18.8, "Available HostnameLookups options"](#). To conserve resources on the server, the default option is **Off**.

Table 18.8. Available HostnameLookups options

Option	Description
On	Enables resolving the IP address for each connection so that the host name 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.

Option	Description
Off	Disables resolving the IP address for each connection.

Note that when the presence of host names 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 18.39. 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 18.40. 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 18.41. 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 18.9, “Available directory listing options”](#). The default options are **Charset=UTF-8**, **FancyIndexing**, **HTMLTable**, **NameWidth=***, and **VersionSort**.

Table 18.9. 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, <i>*</i>) to adjust the width automatically.
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=value	Specifies an icon height. The <i>value</i> is a number of pixels.
IconWidth=value	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=value	Specifies the width of the file name column. The <i>value</i> can be either a number of characters, or an asterisk (that is, <i>*</i>) 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.
SUPPRESSCOLUMN	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.

Option	Description
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 18.42. 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 18.10, “Available KeepAlive options”](#). The default option is **Off**.

Table 18.10. 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 18.43. 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 18.44. 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 18.45. 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 18.46. 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 18.1.6, “Working with Modules”](#) for more information on the Apache HTTP Server’s DSO support.

Example 18.47. 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 18.11, “Common LogFormat options”](#). The *name* can be used instead of the format string in the **CustomLog** directive.

Table 18.11. Common LogFormat options

Option	Description
%b	Represents the size of the response in bytes.
%h	Represents the IP address or host name 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.
%{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 18.48. 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 18.12, “Available LogLevel options”](#). The default option is **warn**.

Table 18.12. 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 18.49. 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 18.50. 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.



IMPORTANT

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 18.51. 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 18.13, “Available server features”](#).

Table 18.13. 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.

Option	Description
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.



IMPORTANT

The **SymLinksIfOwnerMatch** option is not a security feature as it can be bypassed by an attacker.

Example 18.52. 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 18.14, "Available Order options"](#). The default option is **allow,deny**.

Table 18.14. Available Order options

Option	Description
allow,deny	Allow directives are evaluated first.
deny,allow	Deny directives are evaluated first.

Example 18.53. 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 18.54. 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 18.15, “Available ProxyRequests options”](#). The default option is **Off**.

Table 18.15. Available ProxyRequests options

Option	Description
On	Enables forward proxy requests.
Off	Disables forward proxy requests.

Example 18.55. 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 18.56. 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 18.16, "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 18.16. 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 18.57. 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 18.58. 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 18.59. Using the ServerAdmin directive

```
ServerAdmin webmaster@penguin.example.com
```

ServerName

The **ServerName** directive allows you to specify the host name 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 18.60. 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 18.61. 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 18.17, "Available ServerSignature options"](#). The default option is **On**.

Table 18.17. 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 18.62. Using the ServerSignature directive

```
ServerSignature On
```

ServerTokens

The **ServerTokens** directive allows you to customize what information is 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 18.18, "Available ServerTokens options"](#). The default option is **OS**.

Table 18.18. 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).

Option	Description
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 18.63. 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 18.64. 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 18.65. 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 18.66. 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 18.19, "Available UseCanonicalName options"](#). The default option is **Off**.

Table 18.19. 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 host name 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 host name determined by a reverse DNS lookup is used instead.

Example 18.67. 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 18.68. 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 18.20, "Available UserDir options"](#). The default option is **disabled**.

Table 18.20. 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.

NOTE

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:

```
~]# chmod a+x /home/username/
~]# chmod a+rx /home/username/public_html/
```

All files in this directory must be set accordingly.

Example 18.69. Using the UserDir directive

UserDir public_html

18.1.5.2. 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 18.21, "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 18.21. Available SetEnvIf options

Option	Description
Remote_Host	Refers to the client's host name.
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 18.70. 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 18.1.8, "Setting Up an SSL Server"](#) for more information on how to install and configure an SSL server.

18.1.5.3. 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 18.71. 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 18.72. 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 18.73. 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 18.74. 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 18.75. 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 18.76. 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 18.77. 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 18.78. Using the ThreadsPerChild directive

```
ThreadsPerChild 25
```

18.1.6. 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.

18.1.6.1. Loading a Module

To load a particular DSO module, use the **LoadModule** directive as described in [Section 18.1.5.1, “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 18.79. 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 18.1.4.3, “Restarting the Service”](#) for more information on how to restart the **httpd** service.

18.1.6.2. 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**:

```
~]# yum install httpd-devel
```

This package contains the include files, the header files, and the **APache eXtenSion (apxs)** utility required to compile a module.

Once written, you can build the module with the following command:

```
~]# apxs -i -a -c module_name.c
```

If the build was successful, you should be able to load the module the same way as any other module that is distributed with the Apache HTTP Server.

18.1.7. 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, find the virtual host container provided in **/etc/httpd/conf/httpd.conf** as an example, remove the hash sign (that is, **#**) from the beginning of each line, and customize the options according to your requirements as shown in [Example 18.80, "Example virtual host configuration"](#).

Example 18.80. Example virtual host configuration

```
NameVirtualHost *:80

<VirtualHost *:80>
    ServerAdmin webmaster@penguin.example.com
    DocumentRoot /www/docs/penguin.example.com
    ServerName penguin.example.com
    ServerAlias www.penguin.example.com
    ErrorLog logs/penguin.example.com-error_log
    CustomLog logs/penguin.example.com-access_log common
</VirtualHost>
```

Note that **ServerName** must be a valid DNS name assigned to the machine. The **<VirtualHost>** container is highly customizable, and accepts most of the directives available within the main server configuration. Directives that are *not* supported within this container include **User** and **Group**, which were replaced by **SuexecUserGroup**.



NOTE

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 18.1.4.3, "Restarting the Service"](#) for more information on how to restart the **httpd** service.

18.1.8. 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. Red Hat Enterprise Linux also supports the use of Mozilla NSS as the TLS implementation. Support for Mozilla NSS is provided by the **mod_nss** module.

Unlike an HTTP connection that can be read and possibly modified by anybody who is able to intercept

it, the use of SSL/TLS over HTTP, referred to as HTTPS, 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.

18.1.8.1. 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 18.22, “Information about CA lists used by common web browsers”](#).

Table 18.22. Information about CA lists used by common web browsers

Web Browser	Link
Mozilla Firefox	Mozilla root CA list.
Opera	Information on root certificates used by Opera
Internet Explorer	Information on root certificates used by Microsoft Windows
Chromium	Information on root certificates used by the Chromium project

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.

18.1.9. Enabling the mod_ssl Module

If you intend to set up an SSL or HTTPS server using **mod_ssl**, you **cannot** have another application or module, such as **mod_nss** configured to use the same port. Port **443** is the default port for HTTPS.

To set up an SSL server using the **mod_ssl** module and the OpenSSL toolkit, install the `mod_ssl` and `openssl` packages. Enter the following command as **root**:

```
~]# yum install mod_ssl openssl
```

This will create the **mod_ssl** configuration file at `/etc/httpd/conf.d/ssl.conf`, which is included in the main Apache HTTP Server configuration file by default. For the module to be loaded, restart the **httpd** service as described in [Section 18.1.4.3, "Restarting the Service"](#).



IMPORTANT

Due to the vulnerability described in [POODLE: SSLv3 vulnerability \(CVE-2014-3566\)](#), Red Hat recommends disabling **SSL**, if it is enabled, and using only **TLSv1.1** or **TLSv1.2**. Backwards compatibility can be achieved using **TLSv1.0**. Many products Red Hat supports have the ability to use **SSLv2** or **SSLv3** protocols. However, the use of **SSLv2** or **SSLv3** is now strongly recommended against.

18.1.9.1. 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.

Note that in Red Hat Enterprise Linux 6.8 SSLv2 is disabled by default.

Procedure 18.1. 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
# connect. Disable SSLv2 access by default:
SSLProtocol all -SSLv2 -SSLv3
```

Repeat this action for all VirtualHost sections. Save and close the file.

- 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.

- Restart the Apache daemon as follows:

```
~]# service httpd restart
```

Note that any sessions will be interrupted.

Procedure 18.2. Disable All SSL and TLS Protocols Except TLS 1 and Up

To disable all SSL and TLS protocol versions except TLS version 1 and higher, proceed as follows:

- As **root**, open the **/etc/httpd/conf.d/ssl.conf** file and search for **all** instances of **SSLProtocol** directive. By default the 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
```

- Edit the **SSLProtocol** line as follows:

```
# SSL Protocol support:
# List the enable protocol levels with which clients will be able to
# connect. Disable SSLv2 access by default:
SSLProtocol -all +TLSv1 +TLSv1.1 +TLSv1.2
```

Save and close the file.

- Verify the change as follows:

```
~]# grep SSLProtocol /etc/httpd/conf.d/ssl.conf
SSLProtocol -all +TLSv1 +TLSv1.1 +TLSv1.2
```

- Restart the Apache daemon as follows:

```
~]# service httpd restart
```

Note that any sessions will be interrupted.

Procedure 18.3. Testing the Status of SSL and TLS Protocols

To check which versions of SSL and TLS are enabled or disabled, make use of the **openssl s_client -connect** command. The command has the following form:

```
openssl s_client -connect hostname:port -protocol
```

Where *port* is the port to test and *protocol* is the protocol version to test for. To test the SSL server running locally, use **localhost** as the host name. For example, to test the default port for secure HTTPS connections, port **443** to see if SSLv3 is enabled, issue a command as follows:

1. ~]# openssl s_client -connect localhost:443 -ssl3
 CONNECTED(00000003)
 139809943877536:error:14094410:SSL routines:SSL3_READ_BYTES:sslv3 alert
 handshake failure:s3_pkt.c:1257:SSL alert number 40
 139809943877536:error:1409E0E5:SSL routines:SSL3_WRITE_BYTES:ssl handshake
 failure:s3_pkt.c:596:
output omitted
 New, (NONE), Cipher is (NONE)
 Secure Renegotiation IS NOT supported
 Compression: NONE
 Expansion: NONE
 SSL-Session:
 Protocol : SSLv3
output truncated

The above output indicates that the handshake failed and therefore no cipher was negotiated.

2. ~]\$ openssl s_client -connect localhost:443 -tls1_2
 CONNECTED(00000003)
 depth=0 C = --, ST = SomeState, L = SomeCity, O = SomeOrganization, OU =
 SomeOrganizationalUnit, CN = localhost.localdomain, emailAddress =
 root@localhost.localdomain
output omitted
 New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES256-GCM-SHA384
 Server public key is 2048 bit
 Secure Renegotiation IS supported
 Compression: NONE
 Expansion: NONE
 SSL-Session:
 Protocol : TLSv1.2
output truncated

The above output indicates that no failure of the handshake occurred and a set of ciphers was negotiated.

The **openssl s_client** command options are documented in the **s_client(1)** manual page.

For more information on the SSLv3 vulnerability and how to test for it, see the Red Hat Knowledgebase article [POODLE: SSLv3 vulnerability \(CVE-2014-3566\)](#).

18.1.10. Enabling the mod_nss Module

If you intend to set up an HTTPS server using **mod_nss**, the HTTPS server **cannot** simultaneously use **mod_ssl** with its default settings as **mod_ssl** will use port **443** by default, however this is the default HTTPS port. It is recommended to remove the package if it is not required.

To remove **mod_ssl**, enter the following command as **root**:

```
~]# yum remove mod_ssl
```



NOTE

If **mod_ssl** is required for other purposes, modify the **/etc/httpd/conf.d/ssl.conf** file to use a port other than **443** to prevent **mod_ssl** conflicting with **mod_nss** when its port to listen on is changed to **443**.

For a specific VirtualHost where HTTPS is required, **mod_nss** and **mod_ssl** can only co-exist at the same time if they use unique ports. For this reason **mod_nss** by default uses **8443**, but the default port for HTTPS is port **443**. The port is specified by the **Listen** directive as well as in the VirtualHost name or address.

Everything in NSS is associated with a “token”. The software token exists in the NSS database but you can also have a physical token containing certificates. With OpenSSL, discrete certificates and private keys are held in PEM files. With NSS, these are stored in a database. Each certificate and key is associated with a token and each token can have a password protecting it. This password is optional, but if a password is used then the Apache HTTP server needs a copy of it in order to open the database without user intervention at system start.

Procedure 18.4. Configuring mod_nss

1. Install mod_nss as **root**:

```
~]# yum install mod_nss
```

This will create the **mod_nss** configuration file at **/etc/httpd/conf.d/nss.conf**. The **/etc/httpd/conf.d** directory 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 18.1.4.3, “Restarting the Service”](#).

2. As **root**, open the **/etc/httpd/conf.d/nss.conf** file and search for **all** instances of the **Listen** directive.

Edit the **Listen 8443** line as follows:

```
Listen 443
```

Port **443** is the default port for **HTTPS**.

3. Edit the default **VirtualHost _default_:8443** line as follows:

```
VirtualHost _default_:443
```

Edit any other non-default virtual host sections if they exist. Save and close the file.

4. Mozilla NSS stores certificates in a server *certificate database* indicated by the **NSSCertificateDatabase** directive in the **/etc/httpd/conf.d/nss.conf** file. By default the path is set to **/etc/httpd/alias**, the NSS database created during installation.

To view the default NSS database, issue a command as follows:

```
~]# certutil -L -d /etc/httpd/alias
```

Certificate Nickname	Trust Attributes
	SSL,S/MIME,JAR/XPI
cacert	CTu,Cu,Cu
Server-Cert	u,u,u
alpha	u,pu,u

In the above command output, **Server-Cert** is the default **NSSNickname**. The **-L** option lists all the certificates, or displays information about a named certificate, in a certificate database. The **-d** option specifies the database directory containing the certificate and key database files. See the **certutil(1)** man page for more command line options.

- To configure mod_nss to use another database, edit the **NSSCertificateDatabase** line in the **/etc/httpd/conf.d/nss.conf** file. The default file has the following lines within the VirtualHost section.

```
# Server Certificate Database:
# The NSS security database directory that holds the certificates and
# keys. The database consists of 3 files: cert8.db, key3.db and secmod.db.
# Provide the directory that these files exist.
NSSCertificateDatabase /etc/httpd/alias
```

In the above command output, **alias** is the default NSS database directory, **/etc/httpd/alias/**.

- To apply a password to the default NSS certificate database, use the following command as **root**:

```
~]# certutil -W -d /etc/httpd/alias
Enter Password or Pin for "NSS Certificate DB":
Enter a password which will be used to encrypt your keys.
The password should be at least 8 characters long,
and should contain at least one non-alphabetic character.

Enter new password:
Re-enter password:
Password changed successfully.
```

- Before deploying the HTTPS server, create a new certificate database using a certificate signed by a certificate authority (CA).

Example 18.81. Adding a Certificate to the Mozilla NSS database

The **certutil** command is used to add a CA certificate to the NSS database files:

```
certutil -d /etc/httpd/nss-db-directory/ -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 NSS 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. If the database does not exist it will be created. 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.

See the **certutil(1)** man page for more command line options.

8. The NSS database should be password protected to safeguard the private key.

Example 18.82. Setting_a_Password_for_a_Mozilla_NSS_database

The **certutil** tool can be used to set a password for an NSS database as follows:

```
certutil -W -d /etc/httpd/nss-db-directory/
```

For example, for the default database, issue a command as **root** as follows:

```
~]# certutil -W -d /etc/httpd/alias
Enter Password or Pin for "NSS Certificate DB":
Enter a password which will be used to encrypt your keys.
The password should be at least 8 characters long,
and should contain at least one non-alphabetic character.
```

```
Enter new password:
Re-enter password:
Password changed successfully.
```

9. Configure **mod_nss** to use the NSS internal software token by changing the line with the **NSSPassPhraseDialog** directive as follows:

```
~]# vi /etc/httpd/conf.d/nss.conf
NSSPassPhraseDialog file:/etc/httpd/password.conf
```

This is to avoid manual password entry on system start. The software token exists in the NSS database but you can also have a physical token containing your certificates.

10. If the SSL Server Certificate contained in the NSS database is an RSA certificate, make certain that the **NSSNickname** parameter is uncommented and matches the nickname displayed in step 4 above:

```
~]# vi /etc/httpd/conf.d/nss.conf
NSSNickname Server-Cert
```

If the SSL Server Certificate contained in the NSS database is an ECC certificate, make certain that the **NSSECCNickname** parameter is uncommented and matches the nickname displayed in step 4 above:

```
~]# vi /etc/httpd/conf.d/nss.conf
NSSECCNickname Server-Cert
```

Make certain that the **NSSCertificateDatabase** parameter is uncommented and points to the NSS database directory displayed in step 4 or configured in step 5 above:

```
~]# vi /etc/httpd/conf.d/nss.conf
NSSCertificateDatabase /etc/httpd/alias
```

Replace **/etc/httpd/alias** with the path to the certificate database to be used.

11. Create the **/etc/httpd/password.conf** file as **root**:

```
~]# vi /etc/httpd/password.conf
```

Add a line with the following form:

```
internal:password
```

Replacing *password* with the password that was applied to the NSS security databases in step 6 above.

12. Apply the appropriate ownership and permissions to the **/etc/httpd/password.conf** file:

```
~]# chgrp apache /etc/httpd/password.conf
~]# chmod 640 /etc/httpd/password.conf
~]# ls -l /etc/httpd/password.conf
-rw-r-----. 1 root apache 10 Dec 4 17:13 /etc/httpd/password.conf
```

13. To configure **mod_nss** to use the NSS the software token in **/etc/httpd/password.conf**, edit **/etc/httpd/conf.d/nss.conf** as follows:

```
~]# vi /etc/httpd/conf.d/nss.conf
```

14. Restart the Apache server for the changes to take effect as described in [Section 18.1.4.3, "Restarting the Service"](#)



IMPORTANT

Due to the vulnerability described in [POODLE: SSLv3 vulnerability \(CVE-2014-3566\)](#), Red Hat recommends disabling **SSL**, if it is enabled, and using only **TLSv1.1** or **TLSv1.2**. Backwards compatibility can be achieved using **TLSv1.0**. Many products Red Hat supports have the ability to use **SSLv2** or **SSLv3** protocols. However, the use of **SSLv2** or **SSLv3** is now strongly recommended against.

18.1.10.1. Enabling and Disabling SSL and TLS in mod_nss

To disable and enable specific versions of the SSL and TLS protocol, either do it globally by adding the **NSSProtocol** directive in the “## SSL Global Context” section of the configuration file and removing it everywhere else, or edit the default entry under “# SSL Protocol” 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 **NSSProtocol** in the “SSL Global Context” section, or specify it in **all** per-domain VirtualHost sections.

Note that in Red Hat Enterprise Linux 6.8 SSLv2 is disabled by default.

Procedure 18.5. Disable All SSL and TLS Protocols Except TLS 1 and Up in mod_nss

To disable all SSL and TLS protocol versions except TLS version 1 and higher, proceed as follows:

- As **root**, open the **/etc/httpd/conf.d/nss.conf** file and search for **all** instances of the **NSSProtocol** directive. By default, the configuration file contains one section that looks as follows:

```
~]# vi /etc/httpd/conf.d/nss.conf
# SSL Protocol:
output omitted
# Since all protocol ranges are completely inclusive, and no protocol in the
# middle of a range may be excluded, the entry "NSSProtocol SSLv3,TLSv1.1"
# is identical to the entry "NSSProtocol SSLv3,TLSv1.0,TLSv1.1".
NSSProtocol TLSv1.0,TLSv1.1,TLSv1.2
```

This section is within the VirtualHost section.

- Edit the **NSSProtocol** line as follows:

```
# SSL Protocol:
NSSProtocol TLSv1.0,TLSv1.1,TLSv1.2
```

Repeat this action for all VirtualHost sections.

- Edit the **Listen 8443** line as follows:

```
Listen 443
```

- Edit the default **VirtualHost _default_:8443** line as follows:

```
VirtualHost _default_:443
```

Edit any other non-default virtual host sections if they exist. Save and close the file.

- Verify that all occurrences of the **NSSProtocol** directive have been changed as follows:

```
~]# grep NSSProtocol /etc/httpd/conf.d/nss.conf
# middle of a range may be excluded, the entry "NSSProtocol SSLv3,TLSv1.1"
# is identical to the entry "NSSProtocol SSLv3,TLSv1.0,TLSv1.1".
NSSProtocol TLSv1.0,TLSv1.1,TLSv1.2
```

This step is particularly important if you have more than one VirtualHost section.

- Restart the Apache daemon as follows:

```
~]# service httpd restart
```

Note that any sessions will be interrupted.

Procedure 18.6. Testing the Status of SSL and TLS Protocols in mod_nss

To check which versions of SSL and TLS are enabled or disabled in **mod_nss**, make use of the **openssl s_client -connect** command. Install the **openssl** package as **root**:

```
~]# yum install openssl
```

The **openssl s_client -connect** command has the following form:

```
openssl s_client -connect hostname:port -protocol
```

Where *port* is the port to test and *protocol* is the protocol version to test for. To test the SSL server running locally, use **localhost** as the host name. For example, to test the default port for secure HTTPS connections, port **443** to see if SSLv3 is enabled, issue a command as follows:

1. ~]# openssl s_client -connect localhost:443 -ssl3
CONNECTED(00000003)
3077773036:error:1408F10B:SSL routines:SSL3_GET_RECORD:wrong version
number:s3_pkt.c:337:
output omitted
New, (NONE), Cipher is (NONE)
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
SSL-Session:
Protocol : SSLv3
output truncated

The above output indicates that the handshake failed and therefore no cipher was negotiated.

2. ~]\$ openssl s_client -connect localhost:443 -tls1_2
CONNECTED(00000003)
depth=1 C = US, O = example.com, CN = Certificate Shack
output omitted
New, TLSv1/SSLv3, Cipher is AES256-SHA
Server public key is 1024 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
SSL-Session:
Protocol : TLSv1.2
Cipher : AES256-SHA
output truncated

The above output indicates that no failure of the handshake occurred and a set of ciphers was negotiated.

The **openssl s_client** command options are documented in the **s_client(1)** manual page.

For more information on the SSLv3 vulnerability and how to test for it, see the Red Hat Knowledgebase article [POODLE: SSLv3 vulnerability \(CVE-2014-3566\)](#).

18.1.11. 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 18.1.12, “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 18.1.4.3, “Restarting the Service”](#).

Example 18.83. Using a key and certificate from the Red Hat Secure Web Server

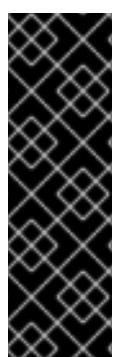
```
~]# mv /etc/httpd/conf/httpsd.key /etc/pki/tls/private/penguin.example.com.key
~]# mv /etc/httpd/conf/httpsd.crt /etc/pki/tls/certs/penguin.example.com.crt
```

18.1.12. 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. In Red Hat Enterprise Linux 6, the **mod_ssl** package is required by the **genkey** utility. To install these, enter the following command as **root**:

```
~]# yum install crypto-utils mod_ssl
```

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.



IMPORTANT

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
```



NOTE

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.

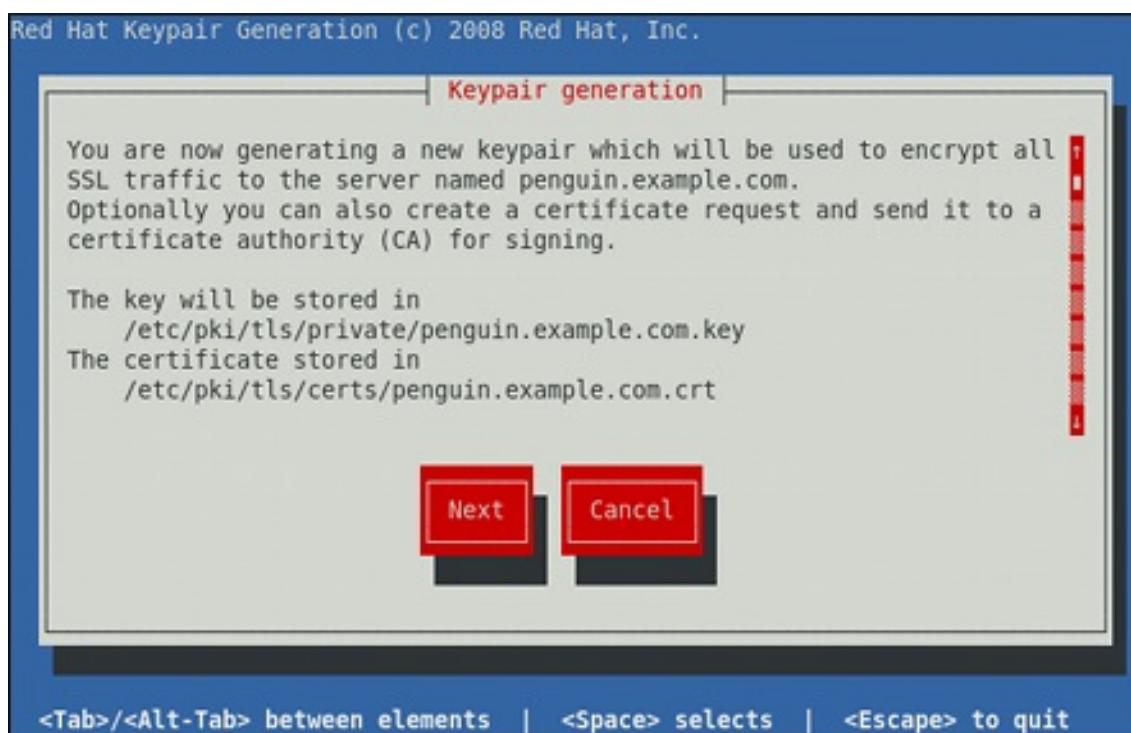


Figure 18.1. Running the **genkey** utility

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](#).

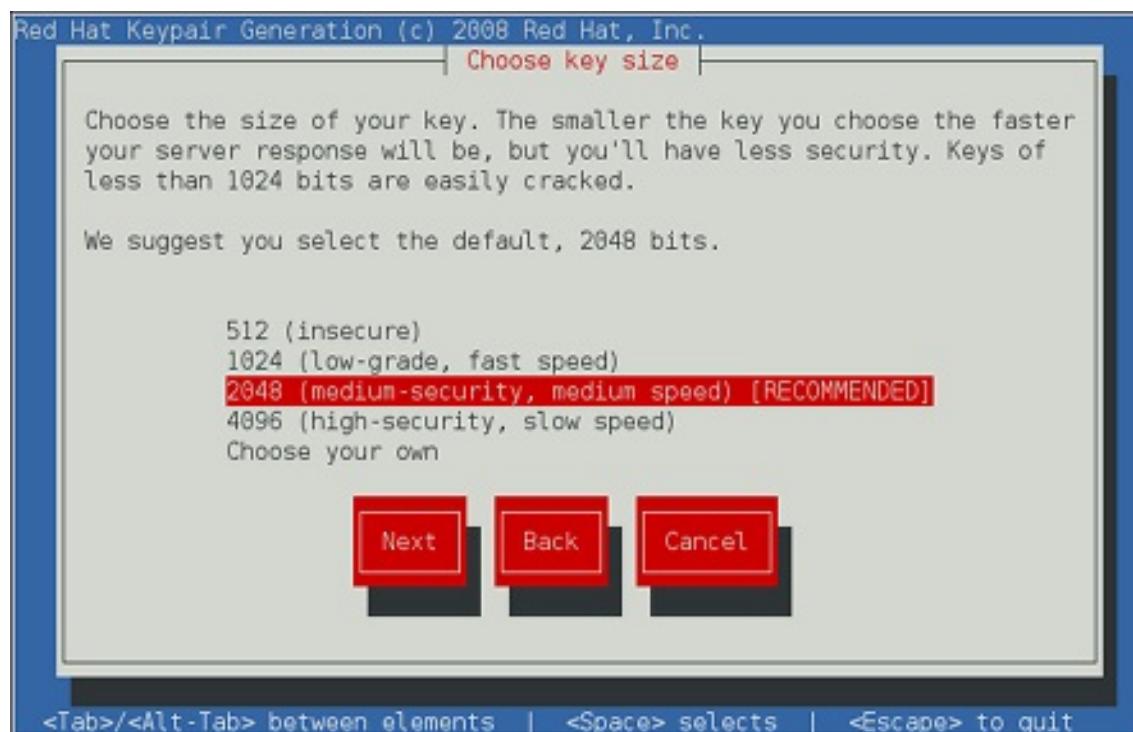


Figure 18.2. Selecting the key size

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.

3. Decide whether you want to send a certificate request to a certificate authority.



Figure 18.3. 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.

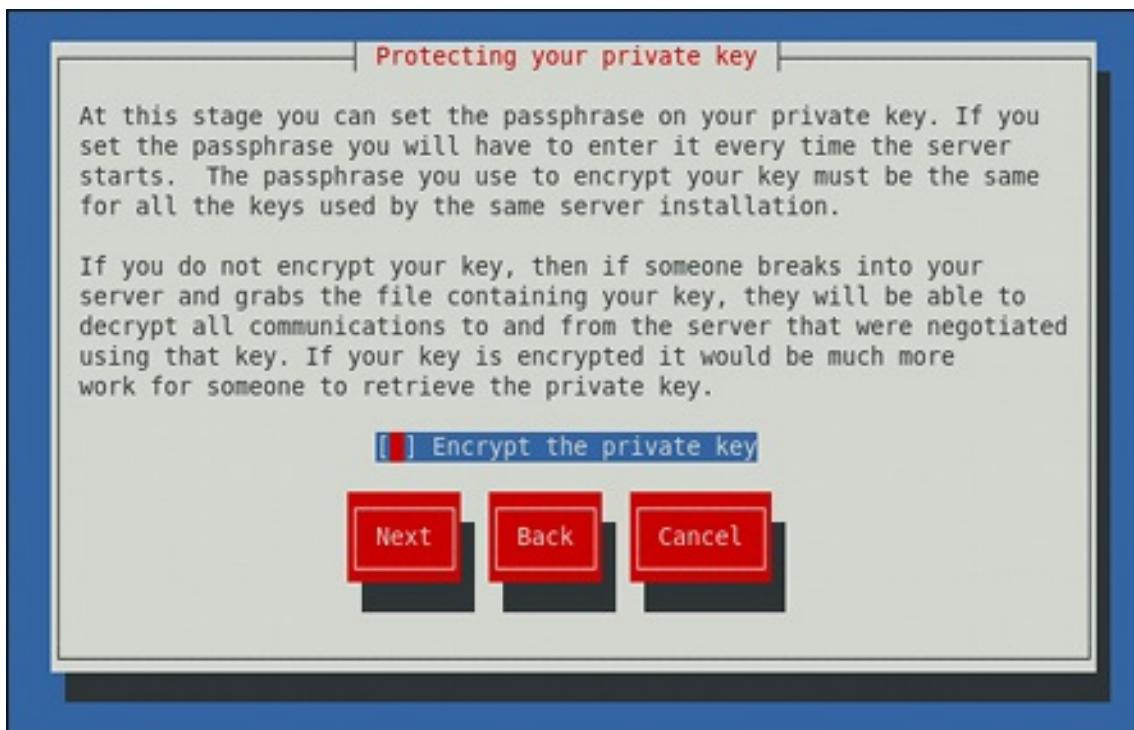


Figure 18.4. Encrypting the private key

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.

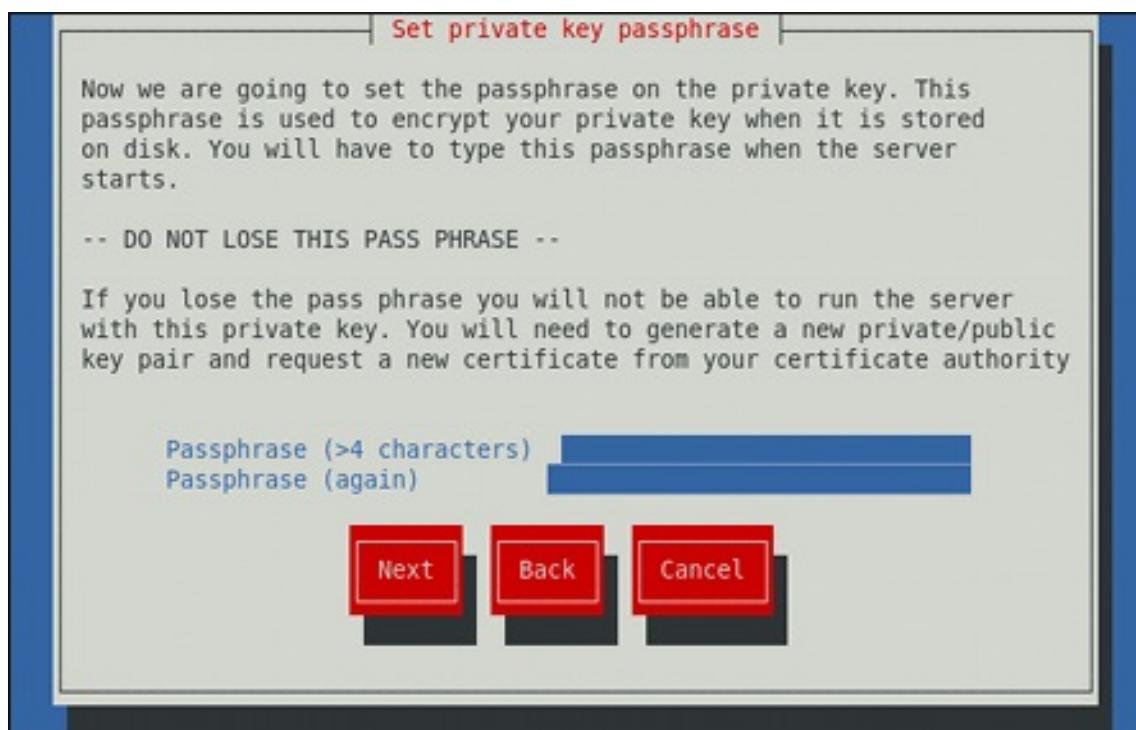
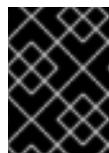


Figure 18.5. Entering a passphrase

Use the **Tab** key to select the **Next** button, and press **Enter** to proceed to the next screen.



IMPORTANT

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.

The screenshot shows a terminal window with a title bar 'Enter details for your certificate'. The window contains instructions: 'You are about to be asked to enter information that will be incorporated into your certificate request to a CA. What you are about to enter is what is called a Distinguished Name or a DN. There are quite a few fields but you can leave some blank.' Below this, there are several input fields with placeholder text and some filled values:

Country Name (ISO 2 letter code)	GB
State or Province Name (full name)	Berkshire
Locality Name (e.g. city)	Newbury
Organization Name (eg, company)	My Company Ltd
Organizational Unit Name (eg, section)	
Common Name (fully qualified domain name)	penguin.example.com
Extra attributes for certificate request:	
Optional challenge password	
Optional company name	

At the bottom of the window are three red buttons labeled 'Next', 'Back', and 'Cancel'.

Figure 18.6. 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-----
MIIBqjCCARMCAQAwajELMAkGA1UEBhMCR0IxEjAQBgNVBAgTCUJlcmtzaGlyZTEQ
MA4GA1UEBxMHTmV3YnVyeTEXMBUGA1UEChMOTXkgQ29tcGFueSBMdGQxHDAaBgNV
BAMTE3Blbmd1aW4uZXhhbXBsZS5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJ
AoGBAJjw8bXq7WKGGXNZsNzltEe9849wUMc4uAh+X8251b8x+ptJQCanGeNhLlxU
xiL5srY2TjoTSQ5DvyFgPQmFFe3cn7v//bKNgNqd4h0EbRFGaj/hDUG3fxnjukX
hP+9iY/eIAQzlHQSkABh/2egtI1lpfDeRvsTUX376TnkIWlhAgMBAAGgADANBgkq
hkIG9w0BAQQFAAOBgQBUTjgjcnts1hZK070c5j+b4IfsBCwm4lnvGx3j0wpLdRq/
rHpx5cbHV99vcKnF3CwDrze9DgpTdldbAccSCVgSG5GE8JZXWYD8EK8p2naJNQl1
YVX1KPi5MPLZuZ9cTb+k4K0cbug0IQiYaKNLNI/0zLE1VEWZXYFX0UBFM2gXYw==
-----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 18.7. 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 18.1.4.3, “Restarting the Service”](#), so that the updated configuration is loaded.

18.1.13. Configure the Firewall for HTTP and HTTPS Using the Command Line

Red Hat Enterprise Linux does not allow **HTTP** and **HTTPS** traffic by default. To enable the system to act as a web server, enable ports and protocols as required. The default port for **HTTP** is **80** and the default port for **HTTPS** is **443**. In both cases the **TCP** should be allowed to pass through the firewall.

To enable port **80** for **HTTP** using the command line, issue the following command as **root**:

```
~]# lokkit --port=80:tcp --update
```

Note that this will restart the firewall as long as it has not been disabled with the **--disabled** option. Active connections will be terminated and time out on the initiating machine. Use the **lokkit --help** command to view the built in help.

To enable port **443** for **HTTPS** using the command line, issue the following command as **root**:

```
~]# lokkit --port=443:tcp --update
```

Note that this will restart the firewall as long as it has not been disabled with the **--disabled** option. Active connections will be terminated and time out on the initiating machine. See the `/etc/services` file for list of services and their associated ports.

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

```
~]# lokkit --update
```

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

```
--port=443:tcp
```

Note that these changes will not take effect even if the firewall is reloaded or the system rebooted. To apply the changes in **/etc/sysconfig/system-config-firewall**, issue the following command as **root**:

```
~]# lokkit --update
```

18.1.13.1. Checking Network Access for Incoming HTTPS and HTTPS Using the Command Line

To check what the firewall is configured to allow, using the command line, issue the following command as **root**:

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

--enabled
--service=ssh
```

In this example taken from a default installation, the firewall is enabled but **HTTP** and **HTTPS** have not been allowed to pass through.

Once the default port for **HTTP** is enabled, the following line appears as output in addition to the lines shown above:

```
--port=80:tcp
```

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

```
~]# iptables -L -n | grep 'tcp.*80'
ACCEPT  tcp  --  0.0.0.0/0      0.0.0.0/0      state NEW tcp dpt:80
```

Once the default port for **HTTPS** is enabled, the following line appears as output in addition to the lines shown above:

```
--port=443:tcp
```

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

```
~]# iptables -L -n | grep 'tcp.*443'  
ACCEPT    tcp  --  0.0.0.0/0            0.0.0.0/0      state NEW tcp dpt:443
```

18.1.14. 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.

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**:

```
~]# yum install httpd-manual  
~]# service httpd graceful
```

Online Documentation

- <http://httpd.apache.org/> – The official website for the Apache HTTP Server with documentation on all the directives and default modules.
- <http://www.openssl.org/> – The OpenSSL home page containing further documentation, frequently asked questions, links to the mailing lists, and other useful resources.

CHAPTER 19. MAIL SERVERS

Red Hat Enterprise Linux offers many advanced applications to serve and access email. This chapter describes modern email protocols in use today, and some of the programs designed to send and receive email.

19.1. 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.

19.1.1. 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)*.

19.1.1.1. SMTP

The primary purpose of SMTP is to transfer email between mail servers. However, it is critical for email clients as well. To send email, the client sends the message to an outgoing mail server, which in turn contacts the destination mail server for delivery. For this reason, it is necessary to specify an SMTP server when configuring an email client.

Under Red Hat Enterprise Linux, a user can configure an SMTP server on the local machine to handle mail delivery. However, it is also possible to configure remote SMTP servers for outgoing mail.

One important point to make about the SMTP protocol is that it does not require authentication. This allows anyone on the Internet to send email to anyone else or even to large groups of people. It is this characteristic of SMTP that makes junk email or *spam* possible. Imposing relay restrictions limits random users on the Internet from sending email through your SMTP server, to other servers on the internet. Servers that do not impose such restrictions are called *open relay* servers.

Red Hat Enterprise Linux provides the Postfix and Sendmail SMTP programs.

19.1.2. 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)*.

19.1.2.1. POP

The default POP server under Red Hat Enterprise Linux is **Dovecot** and is provided by the `dovecot` package.



NOTE

In order to use **Dovecot**, first ensure the dovecot package is installed on your system by running, as **root**:

```
~]# yum install dovecot
```

For more information on installing packages with Yum, see [Section 8.2.4, “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 19.5.1, “Securing Communication”](#).

19.1.2.2. IMAP

The default **IMAP** server under Red Hat Enterprise Linux is **Dovecot** and is provided by the **dovecot** package. See [Section 19.1.2.1, “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 19.5.1, “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.

19.1.2.3. 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:

```
~]# service dovecot restart
```

3. Make the change operational after the next reboot by running the command:

```
~]# chkconfig dovecot on
```



NOTE

Please note that **dovecot** only reports that it started the **IMAP** server, but also starts the **POP3** server.

Unlike **SMTP**, both **IMAP** and **POP3** require connecting clients to authenticate using a user name and password. By default, passwords for both protocols are passed over the network unencrypted.

To configure **SSL** on **dovecot**:

- Edit the **/etc/dovecot/conf.d/10-ssl.conf** configuration to make sure the **ssl_cipher_list** variable is uncommented, and append **:!SSLv3**:

```
ssl_cipher_list = ALL:!LOW:!SSLv2:!EXP:!aNULL:!SSLv3
```

These values ensure that **dovecot** avoids SSL versions 2 and also 3, which are both known to be insecure. This is due to the vulnerability described in [POODLE: SSLv3 vulnerability \(CVE-2014-3566\)](#). See [Resolution for POODLE SSL 3.0 vulnerability \(CVE-2014-3566\) in Postfix and Dovecot](#) for details.

- 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**:

```
~]# service dovecot restart
```

More details on **dovecot** can be found online at <http://www.dovecot.org>.

19.2. 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.

19.2.1. Mail Transport Agent

A *Mail Transport Agent (MTA)* transports email messages between hosts using **SMTP**. A message may involve several MTAs as it moves to its intended destination.

While the delivery of messages between machines may seem rather straightforward, the entire process of deciding if a particular MTA can or should accept a message for delivery is quite complicated. In addition, due to problems from spam, use of a particular MTA is usually restricted by the MTA's configuration or the access configuration for the network on which the MTA resides.

Many modern email client programs can act as an MTA when sending email. However, this action should not be confused with the role of a true MTA. The sole reason email client programs are capable of sending email like an MTA is because the host running the application does not have its own MTA. This is particularly true for email client programs on non-UNIX-based operating systems. However, these client programs only send outbound messages to an MTA they are authorized to use and do not directly deliver the message to the intended recipient's email server.

Since Red Hat Enterprise Linux offers two MTAs, *Postfix* and *Sendmail*, email client programs are often not required to act as an MTA. Red Hat Enterprise Linux also includes a special purpose MTA called *Fetchmail*.

For more information on Postfix, Sendmail, and Fetchmail, see [Section 19.3, "Mail Transport Agents"](#).

19.2.2. 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.

19.2.3. 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 **pine**.

19.3. MAIL TRANSPORT AGENTS

Red Hat Enterprise Linux offers two primary MTAs: Postfix and Sendmail. Postfix is configured as the default MTA, although it is easy to switch the default MTA to Sendmail. To switch the default MTA to Sendmail, you can either uninstall Postfix or use the following command to switch to Sendmail:

```
~]# alternatives --config mta
```

You can also use a command in the following format to enable or disable the desired service:

```
chkconfig service_name on | off
```

19.3.1. 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.

19.3.1.1. The Default Postfix Installation

The Postfix executable is **/usr/sbin/postfix**. This daemon launches all related processes needed to handle mail delivery.

Postfix stores its configuration files in the **/etc/postfix/** directory. The following is a list of the more commonly used files:

- **access** – Used for access control, this file specifies which hosts are allowed to connect to Postfix.
- **main.cf** – The global Postfix configuration file. The majority of configuration options are specified in this file.
- **master.cf** – Specifies how Postfix interacts with various processes to accomplish mail delivery.
- **transport** – Maps email addresses to relay hosts.

The **aliases** file can be found in the `/etc/` directory. This file is shared between Postfix and Sendmail. It is a configurable list required by the mail protocol that describes user ID aliases.



IMPORTANT

The default `/etc/postfix/main.cf` file does not allow Postfix to accept network connections from a host other than the local computer. For instructions on configuring Postfix as a server for other clients, see [Section 19.3.1.2, “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:

```
~]# service postfix restart
```

19.3.1.2. 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/>.

19.3.1.2.1. Configuring Postfix to Use Transport Layer Security

Configuring postfix to use *transport layer security* (TLS) is described in the Red Hat Knowledgebase solution [How to configure postfix with TLS?](#)



IMPORTANT

Due to the vulnerability described in [Resolution for POODLE SSL 3.0 vulnerability \(CVE-2014-3566\) in Postfix and Dovecot](#), Red Hat recommends disabling **SSL**, if it is enabled, and using only **TLSv1.1** or **TLSv1.2**. Backwards compatibility can be achieved using **TLSv1.0**. Many products Red Hat supports have the ability to use **SSLv2** or **SSLv3** protocols. However, the use of **SSLv2** or **SSLv3** is now strongly recommended against.

19.3.1.3. 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.

19.3.1.3.1. The /etc/aliases lookup example

The following is a basic example for using **LDAP** to look up the **/etc/aliases** file. Make sure your **/etc/postfix/main.cf** file contains the following:

```
alias_maps = hash:/etc/aliases, ldap:/etc/postfix/ldap-aliases.cf
```

Create a **/etc/postfix/ldap-aliases.cf** file if you do not have one already and make sure it contains the following:

```
server_host = ldap.example.com
search_base = dc=example, dc=com
```

where **ldap.example.com**, **example**, and **com** are parameters that need to be replaced with specification of an existing available **LDAP** server.



NOTE

The **/etc/postfix/ldap-aliases.cf** file can specify various parameters, including parameters that enable **LDAP SSL** and **STARTTLS**. For more information, see the **ldap_table(5)** man page.

For more information on **LDAP**, see [Section 20.1, “OpenLDAP”](#).

19.3.2. Sendmail

Sendmail's core purpose, like other MTAs, is to safely transfer email among hosts, usually using the **SMTP** protocol. However, Sendmail is highly configurable, allowing control over almost every aspect of how email is handled, including the protocol used. Many system administrators elect to use Sendmail as their MTA due to its power and scalability.

19.3.2.1. 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 19.6, "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)*.

19.3.2.2. The Default Sendmail Installation

In order to use Sendmail, first ensure the sendmail package is installed on your system by running, as **root**:

```
~]# yum install sendmail
```

In order to configure Sendmail, ensure the sendmail-cf package is installed on your system by running, as **root**:

```
~]# yum install sendmail-cf
```

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

Before using Sendmail, the default MTA has to be switched from Postfix. For more information how to switch the default MTA see [Section 19.3, "Mail Transport Agents"](#).

The Sendmail executable is **/usr/sbin/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.

- Alternatively you may use the **m4** macro processor to create a new **/etc/mail/sendmail.cf**. The **m4** macro processor is not installed by default. Before using it to create **/etc/mail/sendmail.cf**, install the m4 package as root:

```
~]# yum install m4
```

More information on configuring Sendmail can be found in [Section 19.3.2.3, “Common Sendmail Configuration Changes”](#).

Various Sendmail configuration files are installed in the `/etc/mail/` directory including:

- **access** – Specifies which systems can use Sendmail for outbound email.
- **domaintable** – Specifies domain name mapping.
- **local-host-names** – Specifies aliases for the host.
- **mailertable** – Specifies instructions that override routing for particular domains.
- **virtusertable** – Specifies a domain-specific form of aliasing, allowing multiple virtual domains to be hosted on one machine.

Several of the configuration files in `/etc/mail/`, such as **access**, **domaintable**, **mailertable** and **virtusertable**, must actually store their information in database files before Sendmail can use any configuration changes. To include any changes made to these configurations in their database files, run the following command, as **root**:

```
~]# makemap hash /etc/mail/<name> < /etc/mail/<name>
```

where `<name>` represents the name of the configuration file to be updated. You may also restart the **sendmail** service for the changes to take effect by running:

```
~]# service sendmail restart
```

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:

```
~]# makemap hash /etc/mail/virtusertable < /etc/mail/virtusertable
```

Sendmail will create an updated **virtusertable.db** file containing the new configuration.

19.3.2.3. Common Sendmail Configuration Changes

When altering the Sendmail configuration file, it is best not to edit an existing file, but to generate an entirely new `/etc/mail/sendmail.cf` file.



WARNING

Before replacing or making any changes to the **sendmail.cf** file, create a backup copy.

To add the desired functionality to Sendmail, edit the **/etc/mail/sendmail.mc** file as root. Once you are finished, restart the **sendmail** service and, if the **m4** package is installed, the **m4** macro processor will automatically generate a new **sendmail.cf** configuration file:

```
~]# service sendmail restart
```



IMPORTANT

The default **sendmail.cf** file does not allow Sendmail to accept network connections from any host other than the local computer. To configure Sendmail as a server for other clients, edit the **/etc/mail/sendmail.mc** file, and either change the address specified in the **Addr=** option of the **DAEMON_OPTIONS** directive from **127.0.0.1** to the IP address of an active network device or comment out the **DAEMON_OPTIONS** directive all together by placing **dnl** at the beginning of the line. When finished, regenerate **/etc/mail/sendmail.cf** by restarting the service

```
~]# service sendmail restart
```

The default configuration in Red Hat Enterprise Linux works for most **SMTP**-only sites. However, it does not work for *UUCP (UNIX-to-UNIX Copy Protocol)* sites. If using UUCP mail transfers, the **/etc/mail/sendmail.mc** file must be reconfigured and a new **/etc/mail/sendmail.cf** file must be generated.

Consult the **/usr/share/sendmail-cf/README** file before editing any files in the directories under the **/usr/share/sendmail-cf** directory, as they can affect the future configuration of the **/etc/mail/sendmail.cf** file.

19.3.2.4. 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**.

19.3.2.5. 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

```
~]# service sendmail restart
```

However users can also be sent spam from servers on the Internet. In these instances, Sendmail's access control features available through the **/etc/mail/access** file can be used to prevent connections from unwanted hosts. The following example illustrates how this file can be used to both block and specifically allow access to the Sendmail server:

```
badspammer.com ERROR:550 "Go away and do not spam us anymore" tux.badspammer.com OK  
10.0 RELAY
```

This example shows that any email sent from **badspammer.com** is blocked with a 550 RFC-821 compliant error code, with a message sent back. Email sent from the **tux.badspammer.com** sub-domain, is accepted. The last line shows that any email sent from the 10.0.*.* network can be relayed through the mail server.

Because the **/etc/mail/access.db** file is a database, use the **makemap** command to update any changes. Do this using the following command as **root**:

```
~]# makemap hash /etc/mail/access < /etc/mail/access
```

Message header analysis allows you to reject mail based on header contents. **SMTP** servers store information about an email's journey in the message header. As the message travels from one MTA to another, each puts in a **Received** header above all the other **Received** headers. It is important to note that this information may be altered by spammers.

The above examples only represent a small part of what Sendmail can do in terms of allowing or blocking access. See the **/usr/share/sendmail-cf/README** file for more information and examples.

Since Sendmail calls the Procmail MDA when delivering mail, it is also possible to use a spam filtering program, such as SpamAssassin, to identify and file spam for users. See [Section 19.4.2.6, “Spam Filters”](#) for more information about using SpamAssassin.

19.3.2.6. 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')dnl  
FEATURE('ldap_routing')dnl
```



NOTE

This is only for a very basic configuration of Sendmail with **LDAP**. The configuration can differ greatly from this depending on the implementation of **LDAP**, especially when configuring several Sendmail machines to use a common **LDAP** server.

Consult **/usr/share/sendmail-cf/README** for detailed **LDAP** routing configuration instructions and examples.

Next, recreate the **/etc/mail/sendmail.cf** file by running the **m4** macro processor and again restarting Sendmail. See [Section 19.3.2.3, “Common Sendmail Configuration Changes”](#) for instructions.

For more information on **LDAP**, see [Section 20.1, “OpenLDAP”](#).

19.3.3. Fetchmail

Fetchmail is an MTA which retrieves email from remote servers and delivers it to the local MTA. Many users appreciate the ability to separate the process of downloading their messages located on a remote server from the process of reading and organizing their email in an MUA. Designed with the needs of dial-up users in mind, Fetchmail connects and quickly downloads all of the email messages to the mail spool file using any number of protocols, including **POP3** and **IMAP**. It can even forward email messages to an **SMTP** server, if necessary.



NOTE

In order to use **Fetchmail**, first ensure the fetchmail package is installed on your system by running, as **root**:

```
~]# yum install fetchmail
```

For more information on installing packages with Yum, see [Section 8.2.4, “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.

19.3.3.1. 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.



NOTE

Users are not required to place their password in the `.fetchmailrc` file. Omitting the **with password '<password>'** section causes Fetchmail to ask for a password when it is launched.

Fetchmail has numerous global, server, and local options. Many of these options are rarely used or only apply to very specific situations. The **fetchmail** man page explains each option in detail, but the most common ones are listed in the following three sections.

19.3.3.2. 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`.

19.3.3.3. 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.

19.3.3.4. 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. At the time of writing, the default action is to use the best available from **SSL2**, **SSL3**, **SSL23**, **TLS1**, **TLS1.1** and **TLS1.2**. Note that **SSL2** is considered obsolete and due to the [POODLE: SSLv3 vulnerability \(CVE-2014-3566\)](#), **SSLv3** should not be used. However there is no way to force the use of TLS1 or newer, therefore ensure the mail server being connected to is configured **not** to use **SSLv2** and **SSLv3**. Use **stunnel** where the server cannot be configured **not** to use **SSLv2** and **SSLv3**.
- **sslproto** – Defines allowed SSL or TLS protocols. Possible values are **SSL2**, **SSL3**, **SSL23**, and **TLS1**. The default value, if **sslproto** is omitted, unset, or set to an invalid value, is **SSL23**. The default action is to use the best from **SSLv3**, **TLSv1**, **TLS1.1** and **TLS1.2**. Note that setting any other value for SSL or TLS will disable all the other protocols. Due to the [POODLE: SSLv3 vulnerability \(CVE-2014-3566\)](#), it is recommended to omit this option, or set it to **SSLv23**, and configure the corresponding mail server **not** to use **SSLv2** and **SSLv3**. Use **stunnel** where the server cannot be configured **not** to use **SSLv2** and **SSLv3**.
- **user "*username*"** – Replace *username* with the username used by Fetchmail to retrieve messages. *This option must precede all other user options.*

19.3.3.5. 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.

19.3.3.6. Informational or Debugging Options

Certain options used after the **fetchmail** command can supply important information.

- **--configdump** – Displays every possible option based on information from **.fetchmailrc** and Fetchmail defaults. No email is retrieved for any users when using this option.
- **-s** – Executes Fetchmail in silent mode, preventing any messages, other than errors, from appearing after the **fetchmail** command.

- **-v** – Executes Fetchmail in verbose mode, displaying every communication between Fetchmail and remote email servers.
- **-V** – Displays detailed version information, lists its global options, and shows settings to be used with each user, including the email protocol and authentication method. No email is retrieved for any users when using this option.

19.3.3.7. Special Options

These options are occasionally useful for overriding defaults often found in the **.fetchmailrc** file.

- **-a** – Fetchmail downloads all messages from the remote email server, whether new or previously viewed. By default, Fetchmail only downloads new messages.
- **-k** – Fetchmail leaves the messages on the remote email server after downloading them. This option overrides the default behavior of deleting messages after downloading them.
- **-l max-number-bytes** – Fetchmail does not download any messages over a particular size and leaves them on the remote email server.
- **--quit** – Quits the Fetchmail daemon process.

More commands and **.fetchmailrc** options can be found in the **fetchmail** man page.

19.3.4. Mail Transport Agent (MTA) Configuration

A *Mail Transport Agent* (MTA) is essential for sending email. A *Mail User Agent* (MUA) such as **Evolution**, **Thunderbird**, or **Mutt**, is used to read and compose email. When a user sends an email from an MUA, the message is handed off to the MTA, which sends the message through a series of MTAs until it reaches its destination.

Even if a user does not plan to send email from the system, some automated tasks or system programs might use the **/bin/mail** command to send email containing log messages to the **root** user of the local system.

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

19.4. MAIL DELIVERY AGENTS

Red Hat Enterprise Linux includes two primary MDAs, Procmail and **mail**. Both of the applications are considered LDAs and both move email from the MTA's spool file into the user's mailbox. However, Procmail provides a robust filtering system.

This section details only Procmail. For information on the **mail** command, consult its man page (**man mail**).

Procmail delivers and filters email as it is placed in the mail spool file of the localhost. It is powerful, gentle on system resources, and widely used. Procmail can play a critical role in delivering email to be read by email client applications.

Procmail can be invoked in several different ways. Whenever an MTA places an email into the mail spool file, Procmail is launched. Procmail then filters and files the email for the MUA and quits. Alternatively, the MUA can be configured to execute Procmail any time a message is received so that messages are

moved into their correct mailboxes. By default, the presence of **/etc/procmailrc** or of a **~/.procmailrc** file (also called an **rc** file) in the user's home directory invokes Procmail whenever an MTA receives a new message.

By default, no system-wide **rc** files exist in the **/etc/** directory and no **.procmailrc** files exist in any user's home directory. Therefore, to use Procmail, each user must construct a **.procmailrc** file with specific environment variables and rules.

Whether Procmail acts upon an email message depends upon whether the message matches a specified set of conditions or *recipes* in the **rc** file. If a message matches a recipe, then the email is placed in a specified file, is deleted, or is otherwise processed.

When Procmail starts, it reads the email message and separates the body from the header information. Next, Procmail looks for a **/etc/procmailrc** file and **rc** files in the **/etc/procmailrcs** directory for default, system-wide, Procmail environmental variables and recipes. Procmail then searches for a **.procmailrc** file in the user's home directory. Many users also create additional **rc** files for Procmail that are referred to within the **.procmailrc** file in their home directory.

19.4.1. Procmail Configuration

The Procmail configuration file contains important environmental variables. These variables specify things such as which messages to sort and what to do with the messages that do not match any recipes.

These environmental variables usually appear at the beginning of the **~/.procmailrc** file in the following format:

```
env-variable="value"
```

In this example, **env-variable** is the name of the variable and **value** defines the variable.

There are many environment variables not used by most Procmail users and many of the more important environment variables are already defined by a default value. Most of the time, the following variables are used:

- **DEFAULT** – Sets the default mailbox where messages that do not match any recipes are placed.

The default **DEFAULT** value is the same as **\$ORGMAIL**.

- **INCLUDERC** – Specifies additional **rc** files containing more recipes for messages to be checked against. This breaks up the Procmail recipe lists into individual files that fulfill different roles, such as blocking spam and managing email lists, that can then be turned off or on by using comment characters in the user's **~/.procmailrc** file.

For example, lines in a user's **~/.procmailrc** file may look like this:

```
MAILDIR=$HOME/Msgs
INCLUDERC=$MAILDIR/lists.rc
INCLUDERC=$MAILDIR/spam.rc
```

To turn off Procmail filtering of email lists but leaving spam control in place, comment out the first **INCLUDERC** line with a hash sign (**#**). Note that it uses paths relative to the current directory.

- **LOCKSLEEP** – Sets the amount of time, in seconds, between attempts by Procmail to use a particular lockfile. The default is **8** seconds.

- **LOCKTIMEOUT** – Sets the amount of time, in seconds, that must pass after a lockfile was last modified before Procmail assumes that the lockfile is old and can be deleted. The default is **1024** seconds.
- **LOGFILE** – The file to which any Procmail information or error messages are written.
- **MAILDIR** – Sets the current working directory for Procmail. If set, all other Procmail paths are relative to this directory.
- **ORGMAIL** – Specifies the original mailbox, or another place to put the messages if they cannot be placed in the default or recipe-required location.

By default, a value of **/var/spool/mail/\$LOGNAME** is used.

- **SUSPEND** – Sets the amount of time, in seconds, that Procmail pauses if a necessary resource, such as swap space, is not available.
- **SWITCHRC** – Allows a user to specify an external file containing additional Procmail recipes, much like the **INCLUDERC** option, except that recipe checking is actually stopped on the referring configuration file and only the recipes on the **SWITCHRC**-specified file are used.
- **VERBOSE** – Causes Procmail to log more information. This option is useful for debugging.

Other important environmental variables are pulled from the shell, such as **LOGNAME**, the login name; **HOME**, the location of the home directory; and **SHELL**, the default shell.

A comprehensive explanation of all environments variables, and their default values, is available in the **procmailrc** man page.

19.4.2. Procmail Recipes

New users often find the construction of recipes the most difficult part of learning to use Procmail. This difficulty is often attributed to recipes matching messages by using *regular expressions* which are used to specify qualifications for string matching. However, regular expressions are not very difficult to construct and even less difficult to understand when read. Additionally, the consistency of the way Procmail recipes are written, regardless of regular expressions, makes it easy to learn by example. To see example Procmail recipes, see [Section 19.4.2.5, “Recipe Examples”](#).

Procmail recipes take the following form:

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

The first two characters in a Procmail recipe are a colon and a zero. Various flags can be placed after the zero to control how Procmail processes the recipe. A colon after the **flags** section specifies that a lockfile is created for this message. If a lockfile is created, the name can be specified by replacing **lockfile-name**.

A recipe can contain several conditions to match against the message. If it has no conditions, every message matches the recipe. Regular expressions are placed in some conditions to facilitate message matching. If multiple conditions are used, they must all match for the action to be performed. Conditions

are checked based on the flags set in the recipe's first line. Optional special characters placed after the asterisk character (*) can further control the condition.

The **action-to-perform** argument specifies the action taken when the message matches one of the conditions. There can only be one action per recipe. In many cases, the name of a mailbox is used here to direct matching messages into that file, effectively sorting the email. Special action characters may also be used before the action is specified. See [Section 19.4.2.4, "Special Conditions and Actions"](#) for more information.

19.4.2.1. Delivering vs. Non-Delivering Recipes

The action used if the recipe matches a particular message determines whether it is considered a *delivering* or *non-delivering* recipe. A delivering recipe contains an action that writes the message to a file, sends the message to another program, or forwards the message to another email address. A non-delivering recipe covers any other actions, such as a *nesting block*. A nesting block is a set of actions, contained in braces { }, that are performed on messages which match the recipe's conditions. Nesting blocks can be nested inside one another, providing greater control for identifying and performing actions on messages.

When messages match a delivering recipe, Procmail performs the specified action and stops comparing the message against any other recipes. Messages that match non-delivering recipes continue to be compared against other recipes.

19.4.2.2. Flags

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

- **A** – Specifies that this recipe is only used if the previous recipe without an **A** or **a** flag also matched this message.
- **a** – Specifies that this recipe is only used if the previous recipe with an **A** or **a** flag also matched this message *and* was successfully completed.
- **B** – Parses the body of the message and looks for matching conditions.
- **b** – Uses the body in any resulting action, such as writing the message to a file or forwarding it. This is the default behavior.
- **c** – Generates a carbon copy of the email. This is useful with delivering recipes, since the required action can be performed on the message and a copy of the message can continue being processed in the **rc** files.
- **D** – Makes the **egrep** comparison case-sensitive. By default, the comparison process is not case-sensitive.
- **E** – While similar to the **A** flag, the conditions in the recipe are only compared to the message if the immediately preceding recipe without an **E** flag did not match. This is comparable to an **else** action.
- **e** – The recipe is compared to the message only if the action specified in the immediately preceding recipe fails.
- **f** – Uses the pipe as a filter.

- **H** – Parses the header of the message and looks for matching conditions. This is the default behavior.
- **h** – Uses the header in a resulting action. This is the default behavior.
- **w** – Tells Procmail to wait for the specified filter or program to finish, and reports whether or not it was successful before considering the message filtered.
- **W** – Is identical to **w** except that "Program failure" messages are suppressed.

For a detailed list of additional flags, see the **procmailrc** man page.

19.4.2.3. Specifying a Local Lockfile

Lockfiles are very useful with Procmail to ensure that more than one process does not try to alter a message simultaneously. Specify a local lockfile by placing a colon (:) after any flags on a recipe's first line. This creates a local lockfile based on the destination file name plus whatever has been set in the **LOCKEXT** global environment variable.

Alternatively, specify the name of the local lockfile to be used with this recipe after the colon.

19.4.2.4. Special Conditions and Actions

Special characters used before Procmail recipe conditions and actions change the way they are interpreted.

The following characters may be used after the asterisk character (*) at the beginning of a recipe's condition line:

- **!** – In the condition line, this character inverts the condition, causing a match to occur only if the condition does not match the message.
- **<** – Checks if the message is under a specified number of bytes.
- **>** – Checks if the message is over a specified number of bytes.

The following characters are used to perform special actions:

- **!** – In the action line, this character tells Procmail to forward the message to the specified email addresses.
- **\$** – Refers to a variable set earlier in the **rc** file. This is often used to set a common mailbox that is referred to by various recipes.
- **|** – Starts a specified program to process the message.
- **{ and }** – Constructs a nesting block, used to contain additional recipes to apply to matching messages.

If no special character is used at the beginning of the action line, Procmail assumes that the action line is specifying the mailbox in which to write the message.

19.4.2.5. Recipe Examples

Procmail is an extremely flexible program, but as a result of this flexibility, composing Procmail recipes from scratch can be difficult for new users.

The best way to develop the skills to build Procmail recipe conditions stems from a strong understanding of regular expressions combined with looking at many examples built by others. A thorough explanation of regular expressions is beyond the scope of this section. The structure of Procmail recipes and useful sample Procmail recipes can be found at various places on the Internet. The proper use and adaptation of regular expressions can be derived by viewing these recipe examples. In addition, introductory information about basic regular expression rules can be found in the **grep(1)** man page.

The following simple examples demonstrate the basic structure of Procmail recipes and can provide the foundation for more intricate constructions.

A basic recipe may not even contain conditions, as is illustrated in the following example:

```
:0:  
new-mail.spool
```

The first line specifies that a local lockfile is to be created but does not specify a name, so Procmail uses the destination file name and appends the value specified in the **LOCKEXT** environment variable. No condition is specified, so every message matches this recipe and is placed in the single spool file called **new-mail.spool**, located within the directory specified by the **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.



WARNING

Be certain that rules are working as intended before sending messages to **/dev/null** for permanent deletion. If a recipe inadvertently catches unintended messages, and those messages disappear, it becomes difficult to troubleshoot the rule.

A better solution is to point the recipe's action to a special mailbox, which can be checked from time to time to look for false positives. Once satisfied that no messages are accidentally being matched, delete the mailbox and direct the action to send the messages to **/dev/null**.

The following recipe grabs email sent from a particular mailing list and places it in a specified folder.

```
:0:  
* ^(From|Cc|To).*tux-lug  
tuxlug
```

Any messages sent from the **tux-lug@domain.com** mailing list are placed in the **tuxlug** mailbox automatically for the MUA. Note that the condition in this example matches the message if it has the mailing list's email address on the **From**, **Cc**, or **To** lines.

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

19.4.2.6. Spam Filters

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

This is particularly true when Procmail is used in conjunction with SpamAssassin. When used together, these two applications can quickly identify spam emails, and sort or destroy them.

SpamAssassin uses header analysis, text analysis, blacklists, a spam-tracking database, and self-learning Bayesian spam analysis to quickly and accurately identify and tag spam.



NOTE

In order to use **SpamAssassin**, first ensure the `spamassassin` package is installed on your system by running, as **root**:

```
~]# yum install spamassassin
```

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

The easiest way for a local user to use SpamAssassin is to place the following line near the top of the `~/.procmailrc` file:

```
INCLUDERC=/etc/mail/spamassassin/spamassassin-default.rc
```

The `/etc/mail/spamassassin/spamassassin-default.rc` contains a simple Procmail rule that activates SpamAssassin for all incoming email. If an email is determined to be spam, it is tagged in the header as such and the title is prepended with the following pattern:

```
*****SPAM*****
```

The message body of the email is also prepended with a running tally of what elements caused it to be diagnosed as spam.

To file email tagged as spam, a rule similar to the following can be used:

```
:0 Hw * ^X-Spam-Status: Yes spam
```

This rule files all email tagged in the header as spam into a mailbox called **spam**.

Since SpamAssassin is a Perl script, it may be necessary on busy servers to use the binary SpamAssassin daemon (**spamd**) and the client application (**spamc**). Configuring SpamAssassin this way, however, requires **root** access to the host.

To start the **spamd** daemon, type the following command:

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

To start the SpamAssassin daemon when the system is booted, use an initscript utility, such as the **Services Configuration Tool (system-config-services)**, to turn on the **spamassassin** service. See [Chapter 12, Services and Daemons](#) for more information about starting and stopping services.

To configure Procmail to use the SpamAssassin client application instead of the Perl script, place the following line near the top of the `~/.procmailrc` file. For a system-wide configuration, place it in `/etc/procmailrc`:

```
INCLUDERC=/etc/mail/spamassassin/spamassassin-spamc.rc
```

19.5. MAIL USER AGENTS

Red Hat Enterprise Linux offers a variety of email programs, both, graphical email client programs, such as **Evolution**, and text-based email programs such as **mutt**.

The remainder of this section focuses on securing communication between a client and a server.

19.5.1. Securing Communication

Popular MUAs included with Red Hat Enterprise Linux, such as **Evolution** and **Mutt** offer SSL-encrypted email sessions.

Like any other service that flows over a network unencrypted, important email information, such as user names, passwords, and entire messages, may be intercepted and viewed by users on the network. Additionally, since the standard **POP** and **IMAP** protocols pass authentication information unencrypted, it is possible for an attacker to gain access to user accounts by collecting user names and passwords as they are passed over the network.

19.5.1.1. Secure Email Clients

Most Linux MUAs designed to check email on remote servers support SSL encryption. To use SSL when retrieving email, it must be enabled on both the email client and the server.

SSL is easy to enable on the client-side, often done with the click of a button in the MUA's configuration window or via an option in the MUA's configuration file. Secure **IMAP** and **POP** have known port numbers (**993** and **995**, respectively) that the MUA uses to authenticate and download messages.

19.5.1.2. Securing Email Client Communications

Offering SSL encryption to **IMAP** and **POP** users on the email server is a simple matter.

First, create an SSL certificate. This can be done in two ways: by applying to a *Certificate Authority (CA)* for an SSL certificate or by creating a self-signed certificate.

**WARNING**

Self-signed certificates should be used for testing purposes only. Any server used in a production environment should use an SSL certificate granted by a CA.

To create a self-signed SSL certificate for **IMAP** or **POP**, change to the **/etc/pki/dovecot** directory, edit the certificate parameters in the **/etc/pki/dovecot/dovecot-openssl.cnf** configuration file as you prefer, and type the following commands, as **root**:

```
dovecot]# rm -f certs/dovecot.pem private/dovecot.pem
dovecot]# /usr/libexec/dovecot/mkcert.sh
```

Once finished, make sure you have the following configurations in your **/etc/dovecot/conf.d/10-ssl.conf** file:

```
ssl_cert = </etc/pki/dovecot/certs/dovecot.pem
ssl_key = </etc/pki/dovecot/private/dovecot.pem
```

Execute the **service dovecot restart** command to restart the **dovecot** daemon.

Alternatively, the **stunnel** command can be used as an encryption wrapper around the standard, non-secure connections to **IMAP** or **POP** services.

The **stunnel** utility uses external OpenSSL libraries included with Red Hat Enterprise Linux to provide strong cryptography and to protect the network connections. It is recommended to apply to a CA to obtain an SSL certificate, but it is also possible to create a self-signed certificate.

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

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

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

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

19.6. ADDITIONAL RESOURCES

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

19.6.1. Installed Documentation

- Information on configuring Sendmail is included with the **sendmail** and **sendmail-cf** packages.

- **/usr/share/sendmail-cf/README** – Contains information on the **m4** macro processor, file locations for Sendmail, supported mailers, how to access enhanced features, and more.

In addition, the **sendmail** and **aliases** man pages contain helpful information covering various Sendmail options and the proper configuration of the Sendmail **/etc/mail/aliases** file.

- **/usr/share/doc/postfix-version-number/** – Contains a large amount of information on how to configure Postfix. Replace *version-number* with the version number of Postfix.
- **/usr/share/doc/fetchmail-version-number** – Contains a full list of Fetchmail features in the **FEATURES** file and an introductory **FAQ** document. Replace *version-number* with the version number of Fetchmail.
- **/usr/share/doc/procmail-version-number/** – Contains a **README** file that provides an overview of Procmail, a **FEATURES** file that explores every program feature, and an **FAQ** file with answers to many common configuration questions. Replace *version-number* with the version number of Procmail.

When learning how Procmail works and creating new recipes, the following Procmail man pages are invaluable:

- **procmail** – Provides an overview of how Procmail works and the steps involved with filtering email.
- **procmailrc** – Explains the **rc** file format used to construct recipes.
- **procmailex** – Gives a number of useful, real-world examples of Procmail recipes.
- **procmailsc** – Explains the weighted scoring technique used by Procmail to match a particular recipe to a message.
- **/usr/share/doc/spamassassin-version-number/** – Contains a large amount of information pertaining to SpamAssassin. Replace *version-number* with the version number of the spamassassin package.

19.6.2. Online Documentation

- [How to configure postfix with TLS?](#) – A Red Hat Knowledgebase article that describes configuring postfix to use TLS.
- The Red Hat Knowledgebase article [How to Configure a System to Manage Multiple Virtual Mailboxes Using Postfix and Dovecot](#) describes managing multiple virtual users under one real-user account using Postfix as Mail Transporting Agent (MTA) and Dovecot as IMAP server.
- <http://www.sendmail.org/> – Offers a thorough technical breakdown of Sendmail features, documentation and configuration examples.
- <http://www.sendmail.com/> – Contains news, interviews and articles concerning Sendmail, including an expanded view of the many options available.
- <http://www.postfix.org/> – The Postfix project home page contains a wealth of information about Postfix. The mailing list is a particularly good place to look for information.
- <http://www.fetchmail.info/fetchmail-FAQ.html> – A thorough FAQ about Fetchmail.
- <http://www.procmail.org/> – The home page for Procmail with links to assorted mailing lists dedicated to Procmail as well as various FAQ documents.

- <http://www.spamassassin.org/> – The official site of the SpamAssassin project.

19.6.3. Related Books

- *Sendmail 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 Procmail, to manage spam problems.
- *Internet Email Protocols: A Developer's Guide* by Kevin Johnson; Addison-Wesley Publishing Company – Provides a very thorough review of major email protocols and the security they provide.
- *Managing IMAP* by Dianna Mullet and Kevin Mullet; O'Reilly & Associates – Details the steps required to configure an IMAP server.

CHAPTER 20. DIRECTORY SERVERS

20.1. OPENLDAP

LDAP (Lightweight Directory Access Protocol) is a set of open protocols used to access centrally stored information over a network. It is based on the **X.500** standard for directory sharing, but is less complex and resource-intensive. For this reason, LDAP is sometimes referred to as "X.500 Lite".

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

LDAP is commonly used for centrally managed users and groups, user authentication, or system configuration. It can also serve as a virtual phone directory, allowing users to easily access contact information for other users. Additionally, it can refer a user to other LDAP servers throughout the world, and thus provide an ad-hoc global repository of information. However, it is most frequently used within individual organizations such as universities, government departments, and private companies.

This section covers the installation and configuration of **OpenLDAP 2.4**, an open source implementation of the LDAPv2 and LDAPv3 protocols.

20.1.1. Introduction to LDAP

Using a client-server architecture, LDAP provides a reliable means to create a central information directory accessible from the network. When a client attempts to modify information within this directory, the server verifies the user has permission to make the change, and then adds or updates the entry as requested. To ensure the communication is secure, the *Transport Layer Security* (TLS) cryptographic protocol can be used to prevent an attacker from intercepting the transmission.



IMPORTANT

The OpenLDAP suite in Red Hat Enterprise Linux 6 no longer uses OpenSSL. Instead, it uses the Mozilla implementation of *Network Security Services* (NSS). OpenLDAP continues to work with existing certificates, keys, and other TLS configuration. For more information on how to configure it to use Mozilla certificate and key database, see [How do I use TLS/SSL with Mozilla NSS](#).



IMPORTANT

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

The LDAP server supports several database systems, which gives administrators the flexibility to choose the best suited solution for the type of information they are planning to serve. Because of a well-defined client *Application Programming Interface* (API), the number of applications able to communicate with an LDAP server is numerous, and increasing in both quantity and quality.

20.1.1.1. LDAP Terminology

The following is a list of LDAP-specific terms that are used within this chapter:

entry

A single unit within an LDAP directory. Each entry is identified by its unique *Distinguished Name* (DN).

attribute

Information directly associated with an entry. For example, if an organization is represented as an LDAP entry, attributes associated with this organization might include an address, a fax number, etc. Similarly, people can be represented as entries with common attributes such as personal telephone number or email address.

An attribute can either have a single value, or an unordered space-separated list of values. While certain attributes are optional, others are required. Required attributes are specified using the **objectClass** definition, and can be found in schema files located in the **/etc/openldap/slapd.d/cn=config/cn=schema** directory.

The assertion of an attribute and its corresponding value is also referred to as a *Relative Distinguished Name* (RDN). Unlike distinguished names that are unique globally, a relative distinguished name is only unique per entry.

LDIF

The *LDAP Data Interchange Format* (LDIF) is a plain text representation of an LDAP entry. It takes the following form:

```
[id] dn: distinguished_name
attribute_type: attribute_value
attribute_type: attribute_value
...
...
```

The optional *id* is a number determined by the application that is used to edit the entry. Each entry can contain as many *attribute_type* and *attribute_value* pairs as needed, as long as they are all defined in a corresponding schema file. A blank line indicates the end of an entry.

20.1.1.2. OpenLDAP Features

OpenLDAP suite provides a number of important features:

- *LDAPv3 Support* – Many of the changes in the protocol since LDAP version 2 are designed to make LDAP more secure. Among other improvements, this includes the support for Simple Authentication and Security Layer (SASL), and Transport Layer Security (TLS) protocols.
- *LDAP Over IPC* – The use of inter-process communication (IPC) enhances security by eliminating the need to communicate over a network.
- *IPv6 Support* – OpenLDAP is compliant with Internet Protocol version 6 (IPv6), the next generation of the Internet Protocol.
- *LDIFv1 Support* – OpenLDAP is fully compliant with LDIF version 1.
- *Updated C API* – The current C API improves the way programmers can connect to and use LDAP directory servers.

- *Enhanced Standalone LDAP Server* – This includes an updated access control system, thread pooling, better tools, and much more.

20.1.1.3. OpenLDAP Server Setup

The typical steps to set up an LDAP server on Red Hat Enterprise Linux are as follows:

1. Install the OpenLDAP suite. See [Section 20.1.2, “Installing the OpenLDAP Suite”](#) for more information on required packages.
2. Customize the configuration as described in [Section 20.1.3, “Configuring an OpenLDAP Server”](#).
3. Start the **slapd** service as described in [Section 20.1.4, “Running an OpenLDAP Server”](#).
4. Use the **ldapadd** utility to add entries to the LDAP directory.
5. Use the **ldapsearch** utility to verify that the **slapd** service is accessing the information correctly.

20.1.2. Installing the OpenLDAP Suite

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

Table 20.1. List of OpenLDAP packages

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

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

Table 20.2. List of commonly installed additional LDAP packages

Package	Description
sssd	A package containing the System Security Services Daemon (SSSD) , a set of daemons to manage access to remote directories and authentication mechanisms. It provides the Name Service Switch (NSS) and the Pluggable Authentication Modules (PAM) interfaces toward the system and a pluggable back-end system to connect to multiple different account sources.

Package	Description
mod_authz_ldap	A package containing mod_authz_ldap , the LDAP authorization module for the Apache HTTP Server. This module uses the short form of the distinguished name for a subject and the issuer of the client SSL certificate to determine the distinguished name of the user within an LDAP directory. It is also capable of authorizing users based on attributes of that user's LDAP directory entry, determining access to assets based on the user and group privileges of the asset, and denying access for users with expired passwords. Note that the mod_ssl module is required when using the mod_authz_ldap module.

To install these packages, use the **yum** command in the following form:

```
yum install package
```

For example, to perform the basic LDAP server installation, type the following at a shell prompt:

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

Note that you must have superuser privileges (that is, you must be logged in as **root**) to run this command. For more information on how to install new packages in Red Hat Enterprise Linux, see [Section 8.2.4, “Installing Packages”](#).

20.1.2.1. Overview of OpenLDAP Server Utilities

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

Table 20.3. List of OpenLDAP server utilities

Command	Description
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 slapd directory based on the current content. Run this utility whenever you change indexing options in the configuration file.

Command	Description
slappasswd	Allows you to create an encrypted user password to be used with the ldapmodify utility, or in the slapd configuration file.
slapschema	Allows you to check the compliance of a database with the corresponding schema.
slaptest	Allows you to check the LDAP server configuration.

For a detailed description of these utilities and their usage, see the corresponding manual pages as referred to in [Section 20.1.6.1, “Installed Documentation”](#).



IMPORTANT

Although only **root** can run **slapadd**, the **slapd** service runs as the **ldap** user. Because of this, the directory server is unable to modify any files created by **slapadd**. To correct this issue, after running the **slapd** utility, type the following at a shell prompt:

```
~]# chown -R ldap:ldap /var/lib/ldap
```



WARNING

To preserve the data integrity, stop the **slapd** service before using **slapadd**, **slapcat**, or **slapindex**. You can do so by typing the following at a shell prompt:

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

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

20.1.2.2. Overview of OpenLDAP Client Utilities

The **openldap-clients** package installs the following utilities which can be used to add, modify, and delete entries in an LDAP directory:

Table 20.4. List of OpenLDAP client utilities

Command	Description
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.

Command	Description
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.

20.1.2.3. Overview of Common LDAP Client Applications

Although there are various graphical LDAP clients capable of creating and modifying directories on the server, none of them is included in Red Hat Enterprise Linux. Popular applications that can access directories in a read-only mode include **Mozilla Thunderbird**, **Evolution**, or **Ekiga**.

20.1.3. Configuring an OpenLDAP Server

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

Table 20.5. List of OpenLDAP configuration files and directories

Path	Description
/etc/openldap/ldap.conf	The configuration file for client applications that use the OpenLDAP libraries. This includes ldapadd , ldapsearch , Evolution , etc.
/etc/openldap/slapd.d/	The directory containing the slapd configuration.

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

```
~]# slapttest -f /etc/openldap/slapd.conf -F /etc/openldap/slapd.d/
```

The **slapd** configuration consists of LDIF entries organized in a hierarchical directory structure, and the recommended way to edit these entries is to use the server utilities described in [Section 20.1.2.1, “Overview of OpenLDAP Server Utilities”](#).



IMPORTANT

An error in an LDIF file can render the **slapd** service unable to start. Because of this, it is strongly advised that you avoid editing the LDIF files within the **/etc/openldap/slapd.d/** directory directly.

20.1.3.1. Changing the Global Configuration

Global configuration options for the LDAP server are stored in the **/etc/openldap/slapd.d/cn=config.ldif** file. The following directives are commonly used:

olcAllows

The **olcAllows** directive allows you to specify which features to enable. It takes the following form:

```
olcAllows: feature
```

It accepts a space-separated list of features as described in [Table 20.6, “Available olcAllows options”](#). The default option is **bind_v2**.

Table 20.6. Available olcAllows options

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

Example 20.1. Using the olcAllows directive

```
olcAllows: bind_v2 update_anon
```

olcConnMaxPending

The **olcConnMaxPending** directive allows you to specify the maximum number of pending requests for an anonymous session. It takes the following form:

```
olcConnMaxPending: number
```

The default option is **100**.

Example 20.2. Using the `olcConnMaxPending` directive

`olcConnMaxPending: 100`

olcConnMaxPendingAuth

The **olcConnMaxPendingAuth** directive allows you to specify the maximum number of pending requests for an authenticated session. It takes the following form:

`olcConnMaxPendingAuth: number`

The default option is **1000**.

Example 20.3. Using the `olcConnMaxPendingAuth` directive

`olcConnMaxPendingAuth: 1000`

olcDisallows

The **olcDisallows** directive allows you to specify which features to disable. It takes the following form:

`olcDisallows: feature`

It accepts a space-separated list of features as described in [Table 20.7, “Available olcDisallows options”](#). No features are disabled by default.

Table 20.7. Available `olcDisallows` options

Option	Description
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 20.4. Using the `olcDisallows` directive

`olcDisallows: bind_anon`

olcIdleTimeout

The **olcIdleTimeout** directive allows you to specify how many seconds to wait before closing an idle connection. It takes the following form:

olcIdleTimeout: *number*

This option is disabled by default (that is, set to **0**).

Example 20.5. Using the olcIdleTimeout directive

olcIdleTimeout: 180

olcLogFile

The **olcLogFile** directive allows you to specify a file in which to write log messages. It takes the following form:

olcLogFile: *file_name*

The log messages are written to standard error by default.

Example 20.6. Using the olcLogFile directive

olcLogFile: /var/log/slapd.log

olcReferral

The **olcReferral** option allows you to specify a URL of a server to process the request in case the server is not able to handle it. It takes the following form:

olcReferral: *URL*

This option is disabled by default.

Example 20.7. Using the olcReferral directive

olcReferral: ldap://root.openldap.org

olcWriteTimeout

The **olcWriteTimeout** option allows you to specify how many seconds to wait before closing a connection with an outstanding write request. It takes the following form:

olcWriteTimeout

This option is disabled by default (that is, set to **0**).

Example 20.8. Using the olcWriteTimeout directive

olcWriteTimeout: 180

20.1.3.2. Changing the Database-Specific Configuration

By default, the OpenLDAP server uses Berkeley DB (BDB) as a database back end. The configuration for this database is stored in the `/etc/openldap/slapd.d/cn=config/olcDatabase={2}bdb.ldif` file. The following directives are commonly used in a database-specific configuration:

olcReadOnly

The **olcReadOnly** directive allows you to use the database in a read-only mode. It takes the following form:

`olcReadOnly: boolean`

It accepts either **TRUE** (enable the read-only mode), or **FALSE** (enable modifications of the database). The default option is **FALSE**.

Example 20.9. Using the olcReadOnly directive

`olcReadOnly: TRUE`

olcRootDN

The **olcRootDN** directive allows you to specify the user that is unrestricted by access controls or administrative limit parameters set for operations on the LDAP directory. It takes the following form:

`olcRootDN: distinguished_name`

It accepts a *Distinguished Name* (DN). The default option is `cn=Manager,dc=my-domain,dc=com`.

Example 20.10. Using the olcRootDN directive

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

olcRootPW

The **olcRootPW** directive allows you to set a password for the user that is specified using the **olcRootDN** directive. It takes the following form:

`olcRootPW: password`

It accepts either a plain text string, or a hash. To generate a hash, type the following at a shell prompt:

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

Example 20.11. Using the olcRootPW directive

```
olcRootPW: {SSHA}WczWsyPEnMchFf1GRTweq2q7XJcvmSxD
```

olcSuffix

The **olcSuffix** directive allows you to specify the domain for which to provide information. It takes the following form:

```
olcSuffix: domain_name
```

It accepts a *fully qualified domain name* (FQDN). The default option is **dc=my-domain,dc=com**.

Example 20.12. Using the olcSuffix directive

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

20.1.3.3. Extending Schema

Since OpenLDAP 2.3, the **/etc/openldap/slapd.d/cn=config/cn=schema** directory also contains LDAP definitions that were previously located in **/etc/openldap/schema**. It is possible to extend the schema used by OpenLDAP to support additional attribute types and object classes using the default schema files as a guide. However, this task is beyond the scope of this chapter. For more information on this topic, see <http://www.openldap.org/doc/admin/schema.html>.

20.1.4. Running an OpenLDAP Server

This section describes how to start, stop, restart, and check the current status of the **Standalone LDAP Daemon**. For more information on how to manage system services in general, see [Chapter 12, Services and Daemons](#).

20.1.4.1. Starting the Service

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

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

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

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

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

20.1.4.2. Stopping the Service

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

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

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

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

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

20.1.4.3. Restarting the Service

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

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

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

20.1.4.4. Checking the Service Status

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

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

20.1.5. Configuring a System to Authenticate Using OpenLDAP

In order to configure a system to authenticate using OpenLDAP, make sure that the appropriate packages are installed on both LDAP server and client machines. For information on how to set up the server, follow the instructions in [Section 20.1.2, “Installing the OpenLDAP Suite”](#) and [Section 20.1.3, “Configuring an OpenLDAP Server”](#). On a client, type the following at a shell prompt:

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

[Chapter 13, Configuring Authentication](#) provides detailed instructions on how to configure applications to use LDAP for authentication.

20.1.5.1. Migrating Old Authentication Information to LDAP Format

The **migrationtools** package provides a set of shell and Perl scripts to help you migrate authentication information into an LDAP format. To install this package, type the following at a shell prompt:

```
~]# yum install migrationtools
```

This will install the scripts to the **/usr/share/migrationtools/** directory. Once installed, edit the **/usr/share/migrationtools/migrate_common.ph** file and change the following lines to reflect the correct domain, for example:

```
# Default DNS domain  
$DEFAULT_MAIL_DOMAIN = "example.com";
```

```
# Default base
$DEFAULT_BASE = "dc=example,dc=com";
```

Alternatively, you can specify the environment variables directly on the command line. For example, to run the **migrate_all_online.sh** script with the default base set to **dc=example,dc=com**, type:

```
~]# export DEFAULT_BASE="dc=example,dc=com" \
/usr/share/migrationtools/migrate_all_online.sh
```

To decide which script to run in order to migrate the user database, see [Table 20.8, “Commonly used LDAP migration scripts”](#).

Table 20.8. Commonly used LDAP migration scripts

Existing Name Service	Is LDAP Running?	Script to Use
/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-version/** directory.

20.1.6. Additional Resources

The following resources offer additional information on the Lightweight Directory Access Protocol. Before configuring LDAP on your system, it is highly recommended that you review these resources, especially the *OpenLDAP Software Administrator's Guide*.

20.1.6.1. Installed Documentation

The following documentation is installed with the **openldap-servers** package:

/usr/share/doc/openldap-servers-version/guide.html

A copy of the *OpenLDAP Software Administrator's Guide*.

/usr/share/doc/openldap-servers-version/README.schema

A README file containing the description of installed schema files.

Additionally, there is also a number of manual pages that are installed with the **openldap**, **openldap-servers**, and **openldap-clients** packages:

Client Applications

- **man ldapadd** – Describes how to add entries to an LDAP directory.
- **man ldapdelete** – Describes how to delete entries within an LDAP directory.
- **man ldapmodify** – Describes how to modify entries within an LDAP directory.
- **man ldapsearch** – Describes how to search for entries within an LDAP directory.
- **man ldappasswd** – Describes how to set or change the password of an LDAP user.
- **man ldapcompare** – Describes how to use the **ldapcompare** tool.
- **man ldapwhoami** – Describes how to use the **ldapwhoami** tool.
- **man ldapmodrdn** – Describes how to modify the RDNs of entries.

Server Applications

- **man slapd** – Describes command-line options for the LDAP server.

Administrative Applications

- **man slapadd** – Describes command-line options used to add entries to a **slapd** database.
- **man slapcat** – Describes command-line options used to generate an LDIF file from a **slapd** database.
- **man slapindex** – Describes command-line options used to regenerate an index based upon the contents of a **slapd** database.
- **man slappasswd** – Describes command-line options used to generate user passwords for LDAP directories.

Configuration Files

- **man ldap.conf** – Describes the format and options available within the configuration file for LDAP clients.
- **man slapd-config** – Describes the format and options available within the configuration directory.

20.1.6.2. Useful Websites

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

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

20.1.6.3. Related Books

OpenLDAP by Example by John Terpstra and Benjamin Coles; Prentice Hall.

A collection of practical exercises in the OpenLDAP deployment.

Implementing LDAP by Mark Wilcox; Wrox Press, Inc.

A book covering LDAP from both the system administrator's and software developer's perspective.

Understanding and Deploying LDAP Directory Services by Tim Howes et al.; Macmillan Technical Publishing.

A book covering LDAP design principles, as well as its deployment in a production environment.

CHAPTER 21. FILE AND PRINT SERVERS

21.1. SAMBA

Samba is the standard open source Windows interoperability suite of programs for Linux. It implements the server message block (**SMB**) protocol. Modern versions of this protocol are also known as the common Internet file system (**CIFS**) protocol. It allows the networking of Microsoft Windows®, Linux, UNIX, and other operating systems together, enabling access to Windows-based file and printer shares. Samba's use of **SMB** allows it to appear as a Windows server to Windows clients.



NOTE

In order to use **Samba**, first ensure the samba package is installed on your system by running the following command as **root**:

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

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

21.1.1. Introduction to Samba

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

What Samba can do:

- Serve directory trees and printers to Linux, UNIX, and Windows clients
- Assist in network browsing (with NetBIOS)
- Authenticate Windows domain logins
- Provide *Windows Internet Name Service* (**WINS**) name server resolution
- Act as a Windows NT®-style *Primary Domain Controller* (PDC)
- Act as a *Backup Domain Controller* (BDC) for a Samba-based PDC
- Act as an Active Directory domain member server
- Join a Windows NT/2000/2003/2008 PDC

What Samba cannot do:

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

21.1.2. Samba Daemons and Related Services

Samba is comprised of three daemons (**smbd**, **nmbd**, and **winbindd**). Three services (**smb**, **nmb**, and **winbind**) control how the daemons are started, stopped, and other service-related features. These

services act as different init scripts. Each daemon is listed in detail below, as well as which specific service has control over it.

smbd

The **smbd** server daemon provides file sharing and printing services to Windows clients. In addition, it is responsible for user authentication, resource locking, and data sharing through the **SMB** protocol. The default ports on which the server listens for **SMB** traffic are **TCP** ports **139** and **445**.

The **smbd** daemon is controlled by the **smb** service.

nmbd

The **nmbd** server daemon understands and replies to NetBIOS name service requests such as those produced by SMB/CIFS in Windows-based systems. These systems include Windows 95/98/ME, Windows NT, Windows 2000, Windows XP, and LanManager clients. It also participates in the browsing protocols that make up the Windows **Network Neighborhood** view. The default port that the server listens to for **NMB** traffic is **UDP** port **137**.

The **nmbd** daemon is controlled by the **nmb** service.

winbindd

The **winbind** service resolves user and group information received from a server running Windows NT, 2000, 2003, Windows Server 2008, or Windows Server 2012. This makes Windows user and group information understandable by UNIX platforms. This is achieved by using Microsoft RPC calls, *Pluggable Authentication Modules* (PAM), and the *Name Service Switch* (NSS). This allows Windows NT domain and Active Directory users to appear and operate as UNIX users on a UNIX machine. Though bundled with the Samba distribution, the **winbind** service is controlled separately from the **smb** service.

The **winbind** daemon is controlled by the **winbind** service and does not require the **smb** service to be started in order to operate. **winbind** is also used when Samba is an Active Directory member, and may also be used on a Samba domain controller (to implement nested groups and interdomain trust). Because **winbind** is a client-side service used to connect to Windows NT-based servers, further discussion of **winbind** is beyond the scope of this chapter.

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



NOTE

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

21.1.3. Connecting to a Samba Share

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

Procedure 21.1. Connecting to a Samba Share Using Nautilus

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

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

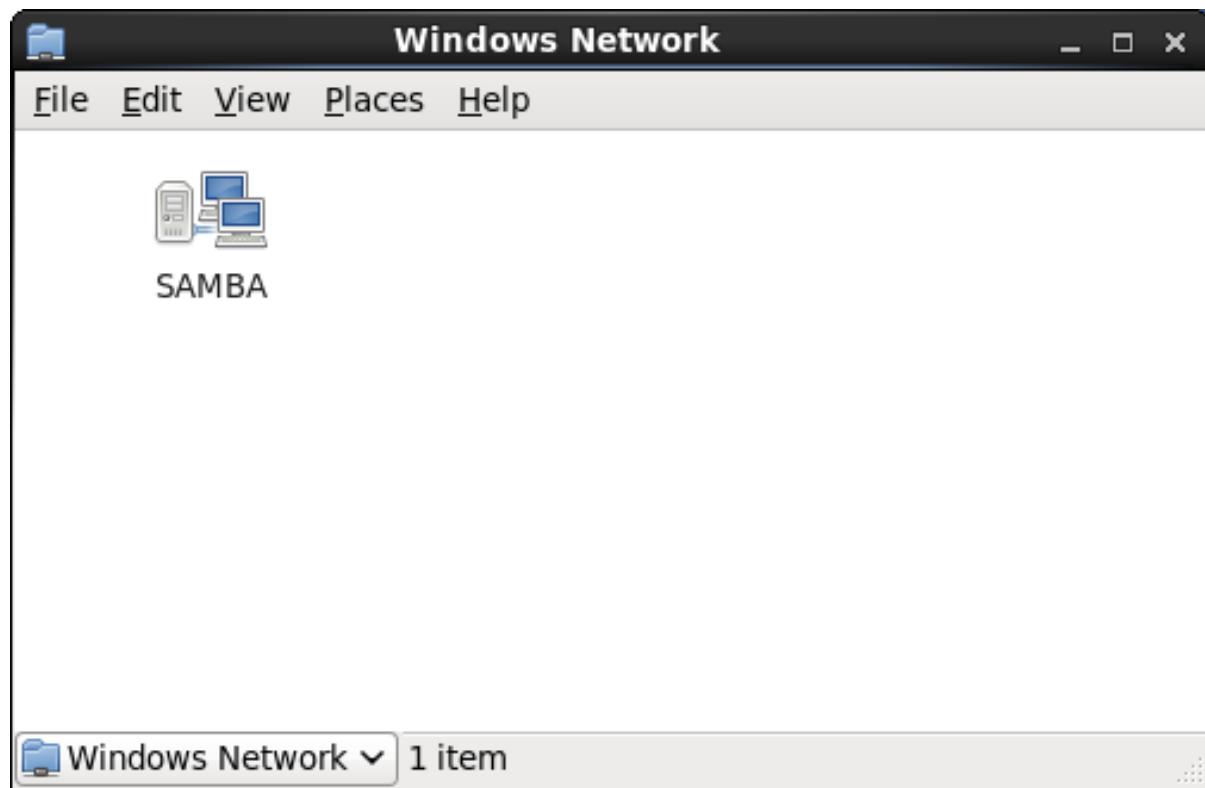


Figure 21.1. SMB Workgroups in Nautilus

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

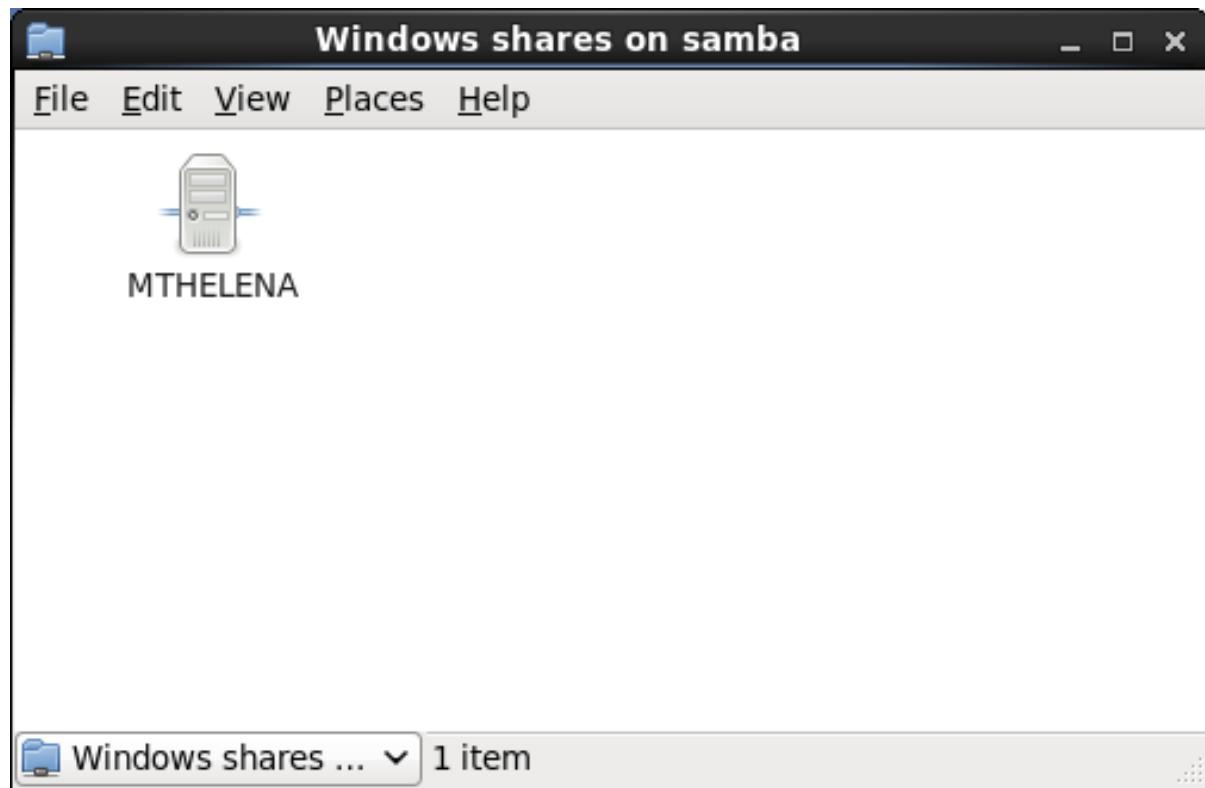


Figure 21.2. SMB Machines in Nautilus

3. As displayed in [Figure 21.2, "SMB Machines in Nautilus"](#), an icon exists for each machine within the workgroup. Double-click on an icon to view the Samba shares on the machine. If a user name and password combination is required, you are prompted for them.

Alternately, you can also specify the Samba server and sharename in the **Location:** bar for **Nautilus** using the following syntax (replace `servername` and `sharename` with the appropriate values):

```
smb://servername/sharename
```

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

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

```
findsmb
```

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

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

Replace `hostname` with the host name or **IP** address of the Samba server you want to connect to, `sharename` with the name of the shared directory you want to browse, and `username` with the Samba user name for the system. Enter the correct password or press **Enter** if no password is required for the user.

If you see the **smb:>** prompt, you have successfully logged in. Once you are logged in, type **help** for a list of commands. If you want to browse the contents of your home directory, replace `sharename` with your user name. If the **-U** switch is not used, the user name of the current user is passed to the Samba server.

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

21.1.3.1. Mounting the Share

Sometimes it is useful to mount a Samba share to a directory so that the files in the directory can be treated as if they are part of the local file system.

To mount a Samba share to a directory, create a directory to mount it to (if it does not already exist), and execute the following command as **root**:

```
mount -t cifs //servername/sharename /mnt/point/ -o  
username=username,password=password
```

This command mounts `sharename` from `servername` in the local directory `/mnt/point/`.

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



NOTE

The **mount.cifs** utility is a separate RPM (independent from Samba). In order to use **mount.cifs**, first ensure the **cifs-utils** package is installed on your system by running the following command as **root**:

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

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

Note that the **cifs-utils** package also contains the **cifs.upcall** binary called by the kernel in order to perform kerberized CIFS mounts. For more information on **cifs.upcall**, see the **cifs.upcall(8)** manual page.



WARNING

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

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

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

21.1.4. Configuring a Samba Server

The default configuration file (**/etc/samba/smb.conf**) allows users to view their home directories as a Samba share. It also shares all printers configured for the system as Samba shared printers. You can attach a printer to the system and print to it from the Windows machines on your network.

21.1.4.1. Graphical Configuration

To configure Samba using a graphical interface, use one of the available Samba graphical user interfaces. A list of available GUIs can be found at <http://www.samba.org/samba/GUI/>.

21.1.4.2. Command-Line Configuration

Samba uses **/etc/samba/smb.conf** as its configuration file. If you change this configuration file, the changes do not take effect until you restart the Samba daemon with the following command as **root**:

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

To specify the Windows workgroup and a brief description of the Samba server, edit the following lines in your **/etc/samba/smb.conf** file:

```
workgroup = WORKGROUPNAME
server string = BRIEF COMMENT ABOUT SERVER
```

Replace **WORKGROUPNAME** with the name of the Windows workgroup to which this machine should belong. The **BRIEF COMMENT ABOUT SERVER** is optional and is used as the Windows comment about the Samba system.

To create a Samba share directory on your Linux system, add the following section to your **/etc/samba/smb.conf** file (after modifying it to reflect your needs and your system):

Example 21.1. An Example Configuration of a Samba Server

```
[sharename]
comment = Insert a comment here
path = /home/share/
valid users = tfox carole
writable = yes
create mask = 0765
```

The above example allows the users **tfox** and **carole** to read and write to the directory **/home/share/**, on the Samba server, from a Samba client.

21.1.4.3. Encrypted Passwords

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

```
smbpasswd -a username
```

21.1.5. Starting and Stopping Samba

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

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



IMPORTANT

To set up a domain member server, you must first join the domain or Active Directory using the **net join** command *before* starting the **smb** service. Also it is recommended to run **winbind** before **smbd**.

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

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

The **restart** option is a quick way of stopping and then starting Samba. This is the most reliable way to make configuration changes take effect after editing the configuration file for Samba. Note that the **restart** option starts the daemon even if it was not running originally.

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

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

The **condrestart** (*conditional restart*) option only stops and starts **smb** on the condition that it is currently running. This option is useful for scripts, because it does not start the daemon if it is not running.



NOTE

When the **/etc/samba/smb.conf** file is changed, Samba automatically reloads it after a few minutes. Issuing a manual **restart** or **reload** is just as effective.

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

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

A manual reload of the **/etc/samba/smb.conf** file can be useful in case of a failed automatic reload by the **smb** service. To ensure that the Samba server configuration file is reloaded without restarting the service, type the following command, as **root**:

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

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

21.1.6. Samba Server Types and the **smb.conf** File

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

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

21.1.6.1. Stand-alone Server

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

Anonymous Read-Only

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

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

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

```

map to guest = Bad user

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

```

Anonymous Read/Write

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



NOTE

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

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

```

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

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

```

Anonymous Print Server

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

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

```

[global]
workgroup = DOCS
netbios name = DOCS_SRV

```

```

security = user
map to guest = Bad user
printing = cups

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

```

Secure Read/Write File and Print Server

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

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

```

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

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

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

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

```

21.1.6.2. Domain Member Server

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

Active Directory Domain Member Server

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

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

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

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

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

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

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

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

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

[domain_realm]
```

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

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

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

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.



NOTE

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

Windows NT4-based Domain Member Server

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

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

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

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

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

Having Samba as a domain member server can be useful in many situations. There are times where the

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



IMPORTANT

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

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

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

21.1.6.3. Domain Controller

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

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



IMPORTANT

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

Primary Domain Controller (PDC) Using **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 21.1.8, “Samba Account Information Databases”](#). The **passdb backend** directive controls which back end is to be used for the PDC.

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

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

```
[global]
workgroup = DOCS
netbios name = DOCS_SRV
passdb backend = tdbsam
security = user
```

```

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

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

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

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

```

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

1. Adjust the **smb.conf** configuration file as shown in [Example 21.7, “An Example Configuration of Primary Domain Controller \(PDC\) Using **tdbsam**”](#).
2. Add the **root** user to the Samba password database. You will be prompted to provide a new Samba password for the **root** user:

```

~]# smbpasswd -a root
New SMB password:

```

3. Start the **smb** service:

```

~]# service smb start

```

4. Make sure all profile, user, and netlogon directories are created.

5. Add groups that users can be members of:

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

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

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

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

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

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

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



NOTE

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

Primary Domain Controller (PDC) with Active Directory

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

21.1.7. Samba Security Modes

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

21.1.7.1. User-Level Security

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

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

```
[GLOBAL]
```

```
...
security = user
...
```

Samba Guest Shares

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

Procedure 21.4. Configuring Samba Guest Shares

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

```
nobody = guest
```

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

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

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

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

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

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

Domain Security Mode (User-Level Security)

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

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

Active Directory Security Mode (User-Level Security)

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

Kerberos tickets.

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

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

21.1.7.2. Share-Level Security

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

21.1.8. Samba Account Information Databases

The following is a list 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.

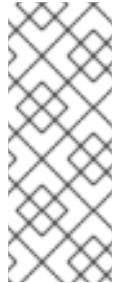
The **tdbsam** back end is recommended for 250 users at most. Larger organizations should require Active Directory or LDAP integration due to scalability and possible network infrastructure concerns.

Idapsam

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

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

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



NOTE

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

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

21.1.9. Samba Network Browsing

Network browsing enables Windows and Samba servers to appear in the Windows **Network Neighborhood**. Inside the **Network Neighborhood**, icons are represented as servers and if opened, the server's shares and printers that are available are displayed.

Network browsing capabilities require NetBIOS over **TCP/IP**. NetBIOS-based networking uses broadcast (**UDP**) messaging to accomplish browse list management. Without NetBIOS and WINS as the primary method for **TCP/IP** host name resolution, other methods such as static files (`/etc/hosts`) or **DNS**, must be used.

A domain master browser collates the browse lists from local master browsers on all subnets so that browsing can occur between workgroups and subnets. Also, the domain master browser should preferably be the local master browser for its own subnet.

21.1.9.1. Domain Browsing

By default, a Windows server PDC for a domain is also the domain master browser for that domain. A Samba server must *not* be set up as a domain master server in this type of situation.

For subnets that do not include the Windows server PDC, a Samba server can be implemented as a local master browser. Configuring the `/etc/samba/smb.conf` file for a local master browser (or no browsing at all) in a domain controller environment is the same as workgroup configuration (see [Section 21.1.4](#),

["Configuring a Samba Server"\).](#)

21.1.9.2. WINS (Windows Internet Name Server)

Either a Samba server or a Windows NT server can function as a WINS server. When a WINS server is used with NetBIOS enabled, UDP unicasts can be routed which allows name resolution across networks. Without a WINS server, the UDP broadcast is limited to the local subnet and therefore cannot be routed to other subnets, workgroups, or domains. If WINS replication is necessary, do not use Samba as your primary WINS server, as Samba does not currently support WINS replication.

In a mixed NT/2000/2003/2008 server and Samba environment, it is recommended that you use the Microsoft WINS capabilities. In a Samba-only environment, it is recommended that you use *only one* Samba server for WINS.

The following is an example of the **/etc/samba/smb.conf** file in which the Samba server is serving as a WINS server:

Example 21.8. An Example Configuration of WINS Server

```
[global]
wins support = yes
```



NOTE

All servers (including Samba) should connect to a WINS server to resolve NetBIOS names. Without WINS, browsing only occurs on the local subnet. Furthermore, even if a domain-wide list is somehow obtained, hosts cannot be resolved for the client without WINS.

21.1.10. Samba with CUPS Printing Support

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

21.1.10.1. Simple smb.conf Settings

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

Example 21.9. An Example Configuration of Samba with CUPS Support

```
[global]
load printers = yes
printing = cups
printcap name = cups
[printers]
comment = All Printers
path = /var/spool/samba
browseable = no
guest ok = yes
writable = no
```

```

printable = yes
printer admin = @ntadmins
[print$]
comment = Printer Drivers Share
path = /var/lib/samba/drivers
write list = ed, john
printer admin = ed, john

```

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

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

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

21.1.11. Samba Distribution Programs

findsmb

findsmb <subnet_broadcast_address>

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

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

```

~]$ findsmb
IP ADDR   NETBIOS NAME WORKGROUP/OS/VERSION
-----
10.1.59.25  VERVE      [MYGROUP] [Unix] [Samba 3.0.0-15]
10.1.59.26  STATION22   [MYGROUP] [Unix] [Samba 3.0.2-7.FC1]
10.1.56.45  TREK        +[WORKGROUP] [Windows 5.0] [Windows 2000 LAN Manager]
10.1.57.94  PIXEL       [MYGROUP] [Unix] [Samba 3.0.0-15]
10.1.57.137 MOBILE001   [WORKGROUP] [Windows 5.0] [Windows 2000 LAN Manager]
10.1.57.141 JAWS        +[KWIKIMART] [Unix] [Samba 2.2.7a-security-rollup-fix]
10.1.56.159 FRED        +[MYGROUP] [Unix] [Samba 3.0.0-14.3E]
10.1.59.192 LEGION     *[MYGROUP] [Unix] [Samba 2.2.7-security-rollup-fix]
10.1.56.205 NANCYN     +[MYGROUP] [Unix] [Samba 2.2.7a-security-rollup-fix]

```

net

net <protocol> <function> <misc_options> <target_options>

The **net** utility is similar to the **net** utility used for Windows and MS-DOS. The first argument is used to specify the protocol to use when executing a command. The **protocol** option can be **ads**, **rap**, or **rpc** for specifying the type of server connection. Active Directory uses **ads**, Win9x/NT3 uses **rap**, and Windows NT4/2000/2003/2008 uses **rpc**. If the protocol is omitted, **net** automatically tries to determine it.

The following example displays a list of the available shares for a host named **wakko**:

```
~]$ net -l share -S wakko
Password:
Enumerating shared resources (exports) on remote server:
Share name  Type   Description
-----
data        Disk   Wakko data share
tmp         Disk   Wakko tmp share
IPC$        IPC    IPC Service (Samba Server)
ADMIN$      IPC    IPC Service (Samba Server)
```

The following example displays a list of Samba users for a host named **wakko**:

```
~]$ net -l user -S wakko
root password:
User name      Comment
-----
andriusb      Documentation
joe           Marketing
lisa          Sales
```

nmblookup

nmblookup <options> <netbios_name>

The **nmblookup** program resolves NetBIOS names into **IP** addresses. The program broadcasts its query on the local subnet until the target machine replies.

The following example displays the **IP** address of the NetBIOS name **trek**:

```
~]$ nmblookup trek
querying trek on 10.1.59.255
10.1.56.45 trek<00>
```

pdbedit

pdbedit <options>

The **pdbedit** program manages accounts located in the SAM database. All back ends are supported including **smbpasswd**, LDAP, and the tdb database library.

The following are examples of adding, deleting, and listing users:

```
~]$ pdbedit -a kristin
new password:
retype new password:
Unix username:    kristin
NT username:
Account Flags:    [U      ]
User SID:        S-1-5-21-1210235352-3804200048-1474496110-2012
Primary Group SID: S-1-5-21-1210235352-3804200048-1474496110-2077
Full Name: Home Directory:  \\wakko\\kristin
HomeDir Drive:
```

```

Logon Script:
Profile Path: \\wakko\kristin\profile
Domain: WAKKO
Account desc:
Workstations: Munged
dial:
Logon time: 0
Logoff time: Mon, 18 Jan 2038 22:14:07 GMT
Kickoff time: Mon, 18 Jan 2038 22:14:07 GMT
Password last set: Thu, 29 Jan 2004 08:29:28
GMT Password can change: Thu, 29 Jan 2004 08:29:28 GMT
Password must change: Mon, 18 Jan 2038 22:14:07 GMT
~]$ pdbedit -v -L kristin
Unix username: kristin
NT username:
Account Flags: [U      ]
User SID: S-1-5-21-1210235352-3804200048-1474496110-2012
Primary Group SID: S-1-5-21-1210235352-3804200048-1474496110-2077
Full Name:
Home Directory: \\wakko\kristin
HomeDir Drive:
Logon Script:
Profile Path: \\wakko\kristin\profile
Domain: WAKKO
Account desc:
Workstations: Munged
dial:
Logon time: 0
Logoff time: Mon, 18 Jan 2038 22:14:07 GMT
Kickoff time: Mon, 18 Jan 2038 22:14:07 GMT
Password last set: Thu, 29 Jan 2004 08:29:28 GMT
Password can change: Thu, 29 Jan 2004 08:29:28 GMT
Password must change: Mon, 18 Jan 2038 22:14:07 GMT
~]$ pdbedit -L
andriusb:505:
joe:503:
lisa:504:
kristin:506:
~]$ pdbedit -x joe
~]$ pdbedit -L
andriusb:505: lisa:504: kristin:506:
```

rpcclient

```
rpcclient <server> <options>
```

The **rpcclient** program issues administrative commands using Microsoft RPCs, which provide access to the Windows administration graphical user interfaces (GUIs) for systems management. This is most often used by advanced users that understand the full complexity of Microsoft RPCs.

smbcacls

```
smbcacls </>server/share> <filename> <options>
```

The **smbcacls** program modifies Windows ACLs on files and directories shared by a Samba server or a Windows server.

smbclient

```
smbclient </>server/share> <password> <options>
```

The **smbclient** program is a versatile UNIX client which provides functionality similar to the **ftp** utility.

smbcontrol

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

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

The **smbcontrol** program sends control messages to running **smbd**, **nmbd**, or **winbindd** daemons. Executing **smbcontrol -i** runs commands interactively until a blank line or a '**q**' is entered.

smbpasswd

```
smbpasswd <options> <username> <password>
```

The **smbpasswd** program manages encrypted passwords. This program can be run by a superuser to change any user's password and also by an ordinary user to change their own Samba password.

smbspool

```
smbspool <job> <user> <title> <copies> <options> <filename>
```

The **smbspool** program is a CUPS-compatible printing interface to Samba. Although designed for use with CUPS printers, **smbspool** can work with non-CUPS printers as well.

smbstatus

```
smbstatus <options>
```

The **smbstatus** program displays the status of current connections to a Samba server.

smbtar

```
smbtar <options>
```

The **smbtar** program performs backup and restores of Windows-based share files and directories to a local tape archive. Though similar to the **tar** utility, the two are not compatible.

testparm

```
testparm <options> <filename> <hostname IP_address>
```

The **testparm** program checks the syntax of the **/etc/samba/smb.conf** file. If your **smb.conf** file is in the default location (**/etc/samba/smb.conf**) you do not need to specify the location. Specifying the host name and **IP** address to the **testparm** program verifies that the **hosts.allow** and **host.deny** files are

configured correctly. The **testparm** program also displays a summary of your **smb.conf** file and the server's role (stand-alone, domain, etc.) after testing. This is convenient when debugging as it excludes comments and concisely presents information for experienced administrators to read. For example:

```
~]$ testparm
Load smb config files from /etc/samba/smb.conf
Processing section "[homes]"
Processing section "[printers]"
Processing section "[tmp]"
Processing section "[html]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions
<enter>
# Global parameters
[global]
workgroup = MYGROUP
server string = Samba Server
security = SHARE
log file = /var/log/samba/%m.log
max log size = 50
socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192
dns proxy = no
[homes]
comment = Home Directories
read only = no
browseable = no
[printers]
comment = All Printers
path = /var/spool/samba
printable = yes
browseable = no
[tmp]
comment = Wakko tmp
path = /tmp
guest only = yes
[html]
comment = Wakko www
path = /var/www/html
force user = andriusb
force group = users
read only = no
guest only = yes
```

wbinfo

wbinfo <options>

The **wbinfo** program displays information from the **winbindd** daemon. The **winbindd** daemon must be running for **wbinfo** to work.

21.1.12. Additional Resources

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

Installed Documentation

- **/usr/share/doc/samba-<version-number>/** – All additional files included with the Samba distribution. This includes all helper scripts, sample configuration files, and documentation.
- See the following man pages for detailed information specific **Samba** features:
 - `smb.conf(5)`
 - `samba(7)`
 - `smbd(8)`
 - `nmbd(8)`
 - `winbindd(8)`

Related Books

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

Useful Websites

- <http://www.samba.org/> – Homepage for the Samba distribution and all official documentation created by the Samba development team. Many resources are available in HTML and PDF formats, while others are only available for purchase. Although many of these links are not Red Hat Enterprise Linux specific, some concepts may apply.
- <http://samba.org/samba/archives.html> – Active email lists for the Samba community. Enabling digest mode is recommended due to high levels of list activity.
- Samba newsgroups – Samba threaded newsgroups, such as www.gmane.org, that use the **NNTP** protocol are also available. This is an alternative to receiving mailing list emails.

21.2. FTP

The *File Transfer Protocol* (**FTP**) is one of the oldest and most commonly used protocols found on the Internet today. Its purpose is to reliably transfer files between computer hosts on a network without requiring the user to log directly in to the remote host or to have knowledge of how to use the remote system. It allows users to access files on remote systems using a standard set of simple commands.

This section outlines the basics of the **FTP** protocol and introduces **vsftpd**, the primary **FTP** server shipped with Red Hat Enterprise Linux.

21.2.1. The File Transfer Protocol

FTP uses a client-server architecture to transfer files using the **TCP** network protocol. Because **FTP** is a

rather old protocol, it uses unencrypted user name and password authentication. For this reason, it is considered an insecure protocol and should not be used unless absolutely necessary. However, because **FTP** is so prevalent on the Internet, it is often required for sharing files to the public. System administrators, therefore, should be aware of **FTP**'s unique characteristics.

This section describes how to configure **vsftpd** to establish connections secured by **TLS** and how to secure an **FTP** server with the help of **SELinux**. A good substitute for **FTP** is **sftp** from the **OpenSSH** suite of tools. For information about configuring **OpenSSH** and about the **SSH** protocol in general, see [Chapter 14, OpenSSH](#).

Unlike most protocols used on the Internet, **FTP** requires multiple network ports to work properly. When an **FTP** client application initiates a connection to an **FTP** server, it opens port **21** on the server – known as the *command port*. This port is used to issue all commands to the server. Any data requested from the server is returned to the client via a *data port*. The port number for data connections, and the way in which data connections are initialized, vary depending upon whether the client requests the data in *active* or *passive* mode.

The following defines these modes:

active mode

Active mode is the original method used by the **FTP** protocol for transferring data to the client application. When an active-mode data transfer is initiated by the **FTP** client, the server opens a connection from port **20** on the server to the **IP** address and a random, unprivileged port (greater than **1024**) specified by the client. This arrangement means that the client machine must be allowed to accept connections over any port above **1024**. With the growth of insecure networks, such as the Internet, the use of firewalls for protecting client machines is now prevalent. Because these client-side firewalls often deny incoming connections from active-mode **FTP** servers, passive mode was devised.

passive mode

Passive mode, like active mode, is initiated by the **FTP** client application. When requesting data from the server, the **FTP** client indicates it wants to access the data in passive mode and the server provides the **IP** address and a random, unprivileged port (greater than **1024**) on the server. The client then connects to that port on the server to download the requested information.

While passive mode does resolve issues for client-side firewall interference with data connections, it can complicate administration of the server-side firewall. You can reduce the number of open ports on a server by limiting the range of unprivileged ports on the **FTP** server. This also simplifies the process of configuring firewall rules for the server. See [Section 21.2.2.6.8, “Network Options”](#) for more information about limiting passive ports.

21.2.2. The **vsftpd** Server

The Very Secure FTP Daemon (**vsftpd**) is designed from the ground up to be fast, stable, and, most importantly, secure. **vsftpd** is the only stand-alone **FTP** server distributed with Red Hat Enterprise Linux, due to its ability to handle large numbers of connections efficiently and securely.

The security model used by **vsftpd** has three primary aspects:

- *Strong separation of privileged and non-privileged processes* – Separate processes handle different tasks, and each of these processes runs with the minimal privileges required for the task.

- Tasks requiring elevated privileges are handled by processes with the minimal privilege necessary – By taking advantage of compatibilities found in the **libcap** library, tasks that usually require full root privileges can be executed more safely from a less privileged process.
- Most processes run in a **chroot** jail – Whenever possible, processes are change-rooted to the directory being shared; this directory is then considered a **chroot** jail. For example, if the **/var/ftp/** directory is the primary shared directory, **vsftpd** reassigns **/var/ftp/** to the new root directory, known as **/**. This disallows any potential malicious hacker activities for any directories not contained in the new root directory.

Use of these security practices has the following effect on how **vsftpd** deals with requests:

- The parent process runs with the least privileges required – The parent process dynamically calculates the level of privileges it requires to minimize the level of risk. Child processes handle direct interaction with the **FTP** clients and run with as close to no privileges as possible.
- All operations requiring elevated privileges are handled by a small parent process – Much like the **Apache HTTP Server**, **vsftpd** launches unprivileged child processes to handle incoming connections. This allows the privileged, parent process to be as small as possible and handle relatively few tasks.
- All requests from unprivileged child processes are distrusted by the parent process – Communication with child processes is received over a socket, and the validity of any information from child processes is checked before being acted on.
- Most interactions with **FTP** clients are handled by unprivileged child processes in a **chroot** jail – Because these child processes are unprivileged and only have access to the directory being shared, any crashed processes only allow the attacker access to the shared files.

21.2.2.1. Starting and Stopping vsftpd

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

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

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

To stop the server, as type:

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

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

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

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

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

To conditionally restart the server, as root type:

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

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

21.2.2.2. Starting Multiple Copies of vsftpd

Sometimes, one computer is used to serve multiple **FTP** domains. This is a technique called *multihoming*. One way to multihome using **vsftpd** is by running multiple copies of the daemon, each with its own configuration file.

To do this, first assign all relevant **IP** addresses to network devices or alias network devices on the system. For more information about configuring network devices, device aliases, see [Chapter 10, NetworkManager](#). For additional information about network configuration scripts, see [Chapter 11, Network Interfaces](#).

Next, the **DNS** server for the **FTP** domains must be configured to reference the correct machine. For information about **BIND**, the **DNS** protocol implementation used in Red Hat Enterprise Linux, and its configuration files, see [Section 17.2, “BIND”](#).

For **vsftpd** to answer requests on different **IP** addresses, multiple copies of the daemon must be running. In order to make this possible, a separate **vsftpd** configuration file for each required instance of the **FTP** server must be created and placed in the **/etc/vsftpd/** directory. Note that each of these configuration files must have a unique name (such as **/etc/vsftpd/vsftpd-site-2.conf**) and must be readable and writable only by the **root** user.

Within each configuration file for each **FTP** server listening on an **IPv4** network, the following directive must be unique:

```
listen_address=N.N.N.N
```

Replace **N.N.N.N** with a *unique IP* address for the **FTP** site being served. If the site is using **IPv6**, use the **listen_address6** directive instead.

Once there are multiple configuration files present in the **/etc/vsftpd/** directory, all configured instances of the **vsftpd** daemon can be started by executing the following command as **root**:

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

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

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

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

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

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

- **anon_root**

- **local_root**
- **vsftpd_log_file**
- **xferlog_file**

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

21.2.2.3. Encrypting vsftpd Connections Using TLS

In order to counter the inherently insecure nature of **FTP**, which transmits user names, passwords, and data without encryption by default, the **vsftpd** daemon can be configured to utilize the **TLS** protocol to authenticate connections and encrypt all transfers. Note that an **FTP** client that supports **TLS** is needed to communicate with **vsftpd** with **TLS** enabled.



NOTE

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

Set the **ssl_enable** configuration directive in the **vsftpd.conf** file to **YES** to turn on **TLS** support. The default settings of other **TLS**-related directives that become automatically active when the **ssl_enable** option is enabled provide for a reasonably well-configured **TLS** set up. This includes, among other things, the requirement to only use the **TLS** v1 protocol for all connections (the use of the insecure **SSL** protocol versions is disabled by default) or forcing all non-anonymous logins to use **TLS** for sending passwords and data transfers.

Example 21.10. Configuring vsftpd to Use TLS

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

```
ssl_enable=YES
ssl_tslv1=YES
ssl_sslv2=NO
ssl_sslv3=NO
```

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

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

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

21.2.2.4. SELinux Policy for vsftpd

The SELinux policy governing the **vsftpd** daemon (as well as other **ftpd** processes), defines a mandatory access control, which, by default, is based on least access required. In order to allow the **FTP** daemon to access specific files or directories, appropriate labels need to be assigned to them.

For example, in order to be able to share files anonymously, the **public_content_t** label must be assigned to the files and directories to be shared. You can do this using the **chcon** command as **root**:

```
~]# chcon -R -t public_content_t /path/to/directory
```

In the above command, replace */path/to/directory* with the path to the directory to which you want to assign the label. Similarly, if you want to set up a directory for uploading files, you need to assign that particular directory the **public_content_rw_t** label. In addition to that, the **allow_ftpd_anon_write** SELinux Boolean option must be set to **1**. Use the **setsebool** command as **root** to do that:

```
~]# setsebool -P allow_ftpd_anon_write=1
```

If you want local users to be able to access their home directories through **FTP**, which is the default setting on Red Hat Enterprise Linux 6, the **ftp_home_dir** Boolean option needs to be set to **1**. If **vsftpd** is to be allowed to run in standalone mode, which is also enabled by default on Red Hat Enterprise Linux 6, the **ftpd_is_daemon** option needs to be set to **1** as well.

See the **ftpd_selinux(8)** manual page for more information, including examples of other useful labels and Boolean options, on how to configure the SELinux policy pertaining to **FTP**. Also, see the [Red Hat Enterprise Linux 6 Security-Enhanced Linux](#) for more detailed information about SELinux in general.

21.2.2.5. Files Installed with vsftpd

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

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

21.2.2.6. vsftpd Configuration Options

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

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

directive=*value*

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



IMPORTANT

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

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

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

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

21.2.2.6.1. Daemon Options

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

- **listen** – When enabled, **vsftpd** runs in standalone mode, which means that the daemon is started independently, not by the **xinetd** super-server. Red Hat Enterprise Linux 6 sets this value to **YES**. Note that the SELinux **ftp_is_daemon** Boolean option needs to be set for **vsftpd** to be allowed to run in standalone mode. See [Section 21.2.2.4, "SELinux Policy for vsftpd"](#) and to [ftp_selinux\(8\)](#) for more information on **vsftpd**'s interaction with the default SELinux policy. This directive cannot be used in conjunction with the **listen_ipv6** directive.

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

- **listen_ipv6** – When enabled, **vsftpd** runs in standalone mode, which means that the daemon is started independently, not by the **xinetd** super-server. With this directive, it only listens on **IPv6** sockets. Note that the SELinux **ftp_is_daemon** Boolean option needs to be set for **vsftpd** to be allowed to run in standalone mode. See [Section 21.2.2.4, "SELinux Policy for vsftpd"](#) and to [ftp_selinux\(8\)](#) for more information on **vsftpd**'s interaction with the default SELinux policy. This directive cannot be used in conjunction with the **listen** directive.

The default value is **NO**.

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

The default value is **NO**.

21.2.2.6.2. Log In Options and Access Controls

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

- **anonymous_enable** – When enabled, anonymous users are allowed to log in. The user names **anonymous** and **ftp** are accepted.

The default value is **YES**.

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

- **banned_email_file** – If the **deny_email_enable** directive is set to **YES**, this directive specifies the file containing a list of anonymous email passwords that are not permitted access to the server.

The default value is **/etc/vsftpd/banned_emails**.

- **banner_file** – Specifies the file containing text displayed when a connection is established to the server. This option overrides any text specified in the **ftpd_banner** directive.

There is no default value for this directive.

- **cmds_allowed** – Specifies a comma-delimited list of **FTP** commands allowed by the server. All other commands are rejected.

There is no default value for this directive.

- **deny_email_enable** – When enabled, any anonymous user utilizing email passwords specified in **/etc/vsftpd/banned_emails** are denied access to the server. The name of the file referenced by this directive can be specified using the **banned_email_file** directive.

The default value is **NO**.

- **ftpd_banner** – When enabled, the string specified within this directive is displayed when a connection is established to the server. This option can be overridden by the **banner_file** directive.

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

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

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

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

- **pam_service_name** – Specifies the PAM service name for **vsftpd**.

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

- **tcp_wrappers** – When enabled, TCP wrappers are used to grant access to the server. If the FTP server is configured on multiple IP addresses, the **VSFTPD_LOAD_CONF** environment variable can be used to load different configuration files based on the IP address being requested by the client.

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

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

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

The default value is **YES**.

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

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

- **userlist_file** – Specifies the file referenced by **vsftpd** when the **userlist_enable** directive is enabled.

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

21.2.2.6.3. Anonymous User Options

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

- **anon_mkdir_write_enable** – When enabled in conjunction with the **write_enable** directive, anonymous users are allowed to create new directories within a parent directory that has write permissions.

The default value is **NO**.

- **anon_root** – Specifies the directory **vsftpd** changes to after an anonymous user logs in.

There is no default value for this directive.

- **anon_upload_enable** – When enabled in conjunction with the **write_enable** directive, anonymous users are allowed to upload files within a parent directory that has write permissions.

The default value is **NO**.

- **anon_world_readable_only** – When enabled, anonymous users are only allowed to download world-readable files.

The default value is **YES**.

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

The default value is **ftp**.

- **no_anon_password** – When enabled, the anonymous user is not asked for a password.

The default value is **NO**.

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

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

The default value is **NO**.

21.2.2.6.4. Local-User Options

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

- **chmod_enable** – When enabled, the **FTP** command **SITE CHMOD** is allowed for local users. This command allows the users to change the permissions on files.

The default value is **YES**.

- **chroot_list_enable** – When enabled, the local users listed in the file specified in the **chroot_list_file** directive are placed in a **chroot** jail upon log in.

If enabled in conjunction with the **chroot_local_user** directive, the local users listed in the file specified in the **chroot_list_file** directive are *not* placed in a **chroot** jail upon log in.

The default value is **NO**.

- **chroot_list_file** – Specifies the file containing a list of local users referenced when the **chroot_list_enable** directive is set to **YES**.

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

- **chroot_local_user** – When enabled, local users are change-rooted to their home directories after logging in.

The default value is **NO**.



WARNING

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

- **guest_enable** – When enabled, all non-anonymous users are logged in as the user **guest**, which is the local user specified in the **guest_username** directive.

The default value is **NO**.

- **guest_username** – Specifies the user name the **guest** user is mapped to.
The default value is **ftp**.
- **local_root** – Specifies the directory **vsftpd** changes to after a local user logs in.
There is no default value for this directive.
- **local_umask** – Specifies the umask value for file creation. Note that the default value is in octal form (a numerical system with a base of eight), which includes a “0” prefix. Otherwise, the value is treated as a base-10 integer.
The default value is **077**. On Red Hat Enterprise Linux 6, this option is set to **022** in the configuration file.
- **passwd_chroot_enable** – When enabled in conjunction with the **chroot_local_user** directive, **vsftpd** change-roots local users based on the occurrence of **./** in the home-directory field within **/etc/passwd**.
The default value is **NO**.
- **user_config_dir** – Specifies the path to a directory containing configuration files bearing the names of local system users that contain specific settings for those users. Any directive in a user's configuration file overrides those found in **/etc/vsftpd/vsftpd.conf**.
There is no default value for this directive.

21.2.2.6.5. Directory Options

The following lists directives that affect directories.

- **dirlist_enable** – When enabled, users are allowed to view directory lists.
The default value is **YES**.
- **dirmessage_enable** – When enabled, a message is displayed whenever a user enters a directory with a message file. This message resides within the current directory. The name of this file is specified in the **message_file** directive and is **.message** by default.
The default value is **NO**. On Red Hat Enterprise Linux 6, this option is set to **YES** in the configuration file.
- **force_dot_files** – When enabled, files beginning with a dot (**.**) are listed in directory listings, with the exception of the **.** and **..** files.
The default value is **NO**.
- **hide_ids** – When enabled, all directory listings show **ftp** as the user and group for each file.
The default value is **NO**.
- **message_file** – Specifies the name of the message file when using the **dirmessage_enable** directive.
The default value is **.message**.

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

The default value is **NO**.

- **use_localtime** – When enabled, directory listings reveal the local time for the computer instead of GMT.

The default value is **NO**.

21.2.2.6.6. File Transfer Options

The following lists directives that affect directories.

- **download_enable** – When enabled, file downloads are permitted.

The default value is **YES**.

- **chown_uploads** – When enabled, all files uploaded by anonymous users are owned by the user specified in the **chown_username** directive.

The default value is **NO**.

- **chown_username** – Specifies the ownership of anonymously uploaded files if the **chown_uploads** directive is enabled.

The default value is **root**.

- **write_enable** – When enabled, **FTP** commands which can change the file system are allowed, such as **DELE**, **RNFR**, and **STOR**.

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

21.2.2.6.7. Logging Options

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

- **dual_log_enable** – When enabled in conjunction with **xferlog_enable**, **vsftpd** writes two files simultaneously: a **wu-ftp**-compatible log to the file specified in the **xferlog_file** directive (**/var/log/xferlog** by default) and a standard **vsftpd** log file specified in the **vsftpd_log_file** directive (**/var/log/vsftpd.log** by default).

The default value is **NO**.

- **log_ftp_protocol** – When enabled in conjunction with **xferlog_enable** and with **xferlog_std_format** set to **NO**, all **FTP** commands and responses are logged. This directive is useful for debugging.

The default value is **NO**.

- **syslog_enable** – When enabled in conjunction with **xferlog_enable**, all logging normally written to the standard **vsftpd** log file specified in the **vsftpd_log_file** directive (**/var/log/vsftpd.log** by default) is sent to the system logger instead under the **FTPD** facility.

The default value is **NO**.

- **vsftpd_log_file** – Specifies the **vsftpd** log file. For this file to be used, **xferlog_enable** must be enabled and **xferlog_std_format** must either be set to **NO** or, if **xferlog_std_format** is set to **YES**, **dual_log_enable** must be enabled. It is important to note that if **syslog_enable** is set to **YES**, the system log is used instead of the file specified in this directive.

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

- **xferlog_enable** – When enabled, **vsftpd** logs connections (**vsftpd** format only) and file-transfer information to the log file specified in the **vsftpd_log_file** directive (**/var/log/vsftpd.log** by default). If **xferlog_std_format** is set to **YES**, file-transfer information is logged, but connections are not, and the log file specified in **xferlog_file** (**/var/log/xferlog** by default) is used instead. It is important to note that both log files and log formats are used if **dual_log_enable** is set to **YES**.

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

- **xferlog_file** – Specifies the **wu-ftp**-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-ftp**-compatible file-transfer log is written to the file specified in the **xferlog_file** directive (**/var/log/xferlog** by default). It is important to note that this file only logs file transfers and does not log connections to the server.

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



IMPORTANT

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

21.2.2.6.8. Network Options

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

- **accept_timeout** – Specifies the amount of time for a client using passive mode to establish a connection.

The default value is **60**.

- **anon_max_rate** – Specifies the maximum data transfer rate for anonymous users in bytes per second.

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

- **connect_from_port_20** – When enabled, **vsftpd** runs with enough privileges to open port **20** on the server during active-mode data transfers. Disabling this option allows **vsftpd** to run with less privileges but may be incompatible with some **FTP** clients.

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

- **connect_timeout** – Specifies the maximum amount of time a client using active mode has to respond to a data connection, in seconds.

The default value is **60**.

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

The default value is **300**.

- **ftp_data_port** – Specifies the port used for active data connections when **connect_from_port_20** is set to **YES**.

The default value is **20**.

- **idle_session_timeout** – Specifies the maximum amount of time between commands from a remote client. Once triggered, the connection to the remote client is closed.

The default value is **300**.

- **listen_address** – Specifies the **IP** address on which **vsftpd** listens for network connections.

There is no default value for this directive.



NOTE

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

- **listen_address6** – Specifies the **IPv6** address on which **vsftpd** listens for network connections when **listen_ipv6** is set to **YES**.

There is no default value for this directive.



NOTE

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

- **listen_port** – Specifies the port on which **vsftpd** listens for network connections.

The default value is **21**.

- **local_max_rate** – Specifies the maximum rate at which data is transferred for local users logged in to the server in bytes per second.

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

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

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

- **max_per_ip** – Specifies the maximum number of clients allowed to connect from the same source **IP** address.

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

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

There is no default value for this directive.

- **pasv_enable** – When enabled, passive-mode connections are allowed.

The default value is **YES**.

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

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

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

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

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



WARNING

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

The default value is **NO**.

- **port_enable** – When enabled, active-mode connects are allowed.

The default value is **YES**.

21.2.2.6.9. Security Options

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

- **isolate_network** – If enabled, **vsftpd** uses the **CLONE_NEWWNET** container flag to isolate the unprivileged protocol handler processes, so that they cannot arbitrarily call **connect()** and instead have to ask the privileged process for sockets (the **port_promiscuous** option must be disabled).

The default value is **YES**.

- **isolate** – If enabled, **vsftpd** uses the **CLONE_NEWPID** and **CLONE_NEWIPC** container flags to isolate processes to their IPC and PID namespaces to prevent them from interacting with each other.

The default value is **YES**.

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

The default value is **NO**.

- **allow_anon_ssl** – Specifies whether anonymous users should be allowed to use secured **SSL** connections.

The default value is **NO**.

- **require_cert** – If enabled, all **SSL** client connections are required to present a client certificate.

The default value is **NO**.

21.2.3. Additional Resources

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

21.2.3.1. Installed Documentation

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

Server Applications

- **vsftpd(8)** – Describes available command-line options for **vsftpd**.

Configuration Files

- **vsftpd.conf(5)** – Contains a detailed list of options available within the configuration file for **vsftpd**.

- hosts_access(5) – Describes the format and options available within the **TCP** wrappers configuration files: **hosts.allow** and **hosts.deny**.

Interaction with SELinux

- man ftpd_selinux** – Contains a description of the *SELinux* policy governing **ftpd** processes as well as an explanation of the way SELinux labels need to be assigned and Booleans set.

21.2.3.2. Online Documentation

About vsftpd and FTP in General

- <http://vsftpd.beasts.org/> – The **vsftpd** project page is a great place to locate the latest documentation and to contact the author of the software.
- <http://slacksite.com/other/ftp.html> – This website provides a concise explanation of the differences between active and passive-mode **FTP**.

Red Hat Enterprise Linux Documentation

- [Red Hat Enterprise Linux 6 Security-Enhanced Linux](#) – The *Security-Enhanced Linux* for Red Hat Enterprise Linux 6 describes the basic principles of SELinux and documents in detail how to configure and use SELinux with various services such as the Apache HTTP Server, Postfix, PostgreSQL, or OpenShift. It explains how to configure SELinux access permissions for system services managed by systemd.
- [Red Hat Enterprise Linux 6 Security Guide](#) – The *Security Guide* for Red Hat Enterprise Linux 6 assists users and administrators in learning the processes and practices of securing their workstations and servers against local and remote intrusion, exploitation, and malicious activity. It also explains how to secure critical system services.

Relevant RFC Documents

- [RFC 0959](#) – The original *Request for Comments* (RFC) of the **FTP** protocol from the IETF.
- [RFC 1123](#) – The small **FTP**-related section extends and clarifies RFC 0959.
- [RFC 2228](#) – **FTP** security extensions. **vsftpd** implements the small subset needed to support TLS and SSL connections.
- [RFC 2389](#) – Proposes **FEAT** and **OPTS** commands.
- [RFC 2428](#) – **IPv6** support.

21.3. PRINTER CONFIGURATION

The **Printer Configuration** tool serves for printer configuring, maintenance of printer configuration files, print spool directories and print filters, and printer classes management.

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



IMPORTANT

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

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



NOTE

You can perform the same and additional operations on printers directly from the CUPS web application or command line. To access the application, in a web browser, go to <http://localhost:631/>. For CUPS manuals see the links on the **Home** tab of the web site.

21.3.1. Starting the Printer Configuration Tool

With the Printer Configuration tool you can perform various operations on existing printers and set up new printers. However, you can use also CUPS directly (go to <http://localhost:631/> to access CUPS).

On the panel, click **System → Administration → Printing**, or run the **system-config-printer** command from the command line to start the tool.

The **Printer Configuration** window depicted in [Figure 21.3, "Printer Configuration window"](#) appears.

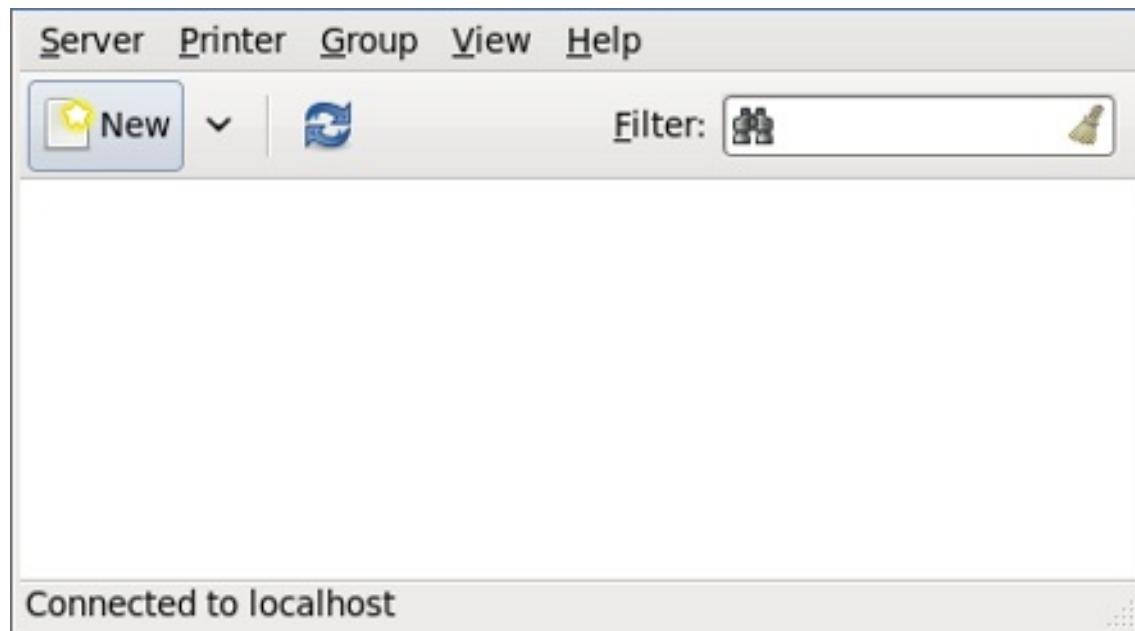


Figure 21.3. Printer Configuration window

21.3.2. Starting Printer Setup

Printer setup process varies depending on the printer queue type.

If you are setting up a local printer connected with USB, the printer is discovered and added automatically. You will be prompted to confirm the packages to be installed and provide the root password. Local printers connected with other port types and network printers need to be set up manually.

Follow this procedure to start a manual printer setup:

1. Start the Printer Configuration tool (see [Section 21.3.1, "Starting the Printer Configuration Tool"](#)).
2. Go to **Server → New → Printer**.
3. In the **Authenticate** dialog box, type the root user password and confirm.
4. Select the printer connection type and provide its details in the area on the right.

21.3.3. Adding a Local Printer

Follow this procedure to add a local printer connected with other than a serial port:

1. Open the **New Printer** dialog (see [Section 21.3.2, "Starting Printer Setup"](#)).
2. If the device does not appear automatically, select the port to which the printer is connected in the list on the left (such as **Serial Port #1** or **LPT #1**).
3. On the right, enter the connection properties:

for Other

URI (for example file:/dev/lp0)

for Serial Port

Baud Rate

Parity

Data Bits

Flow Control

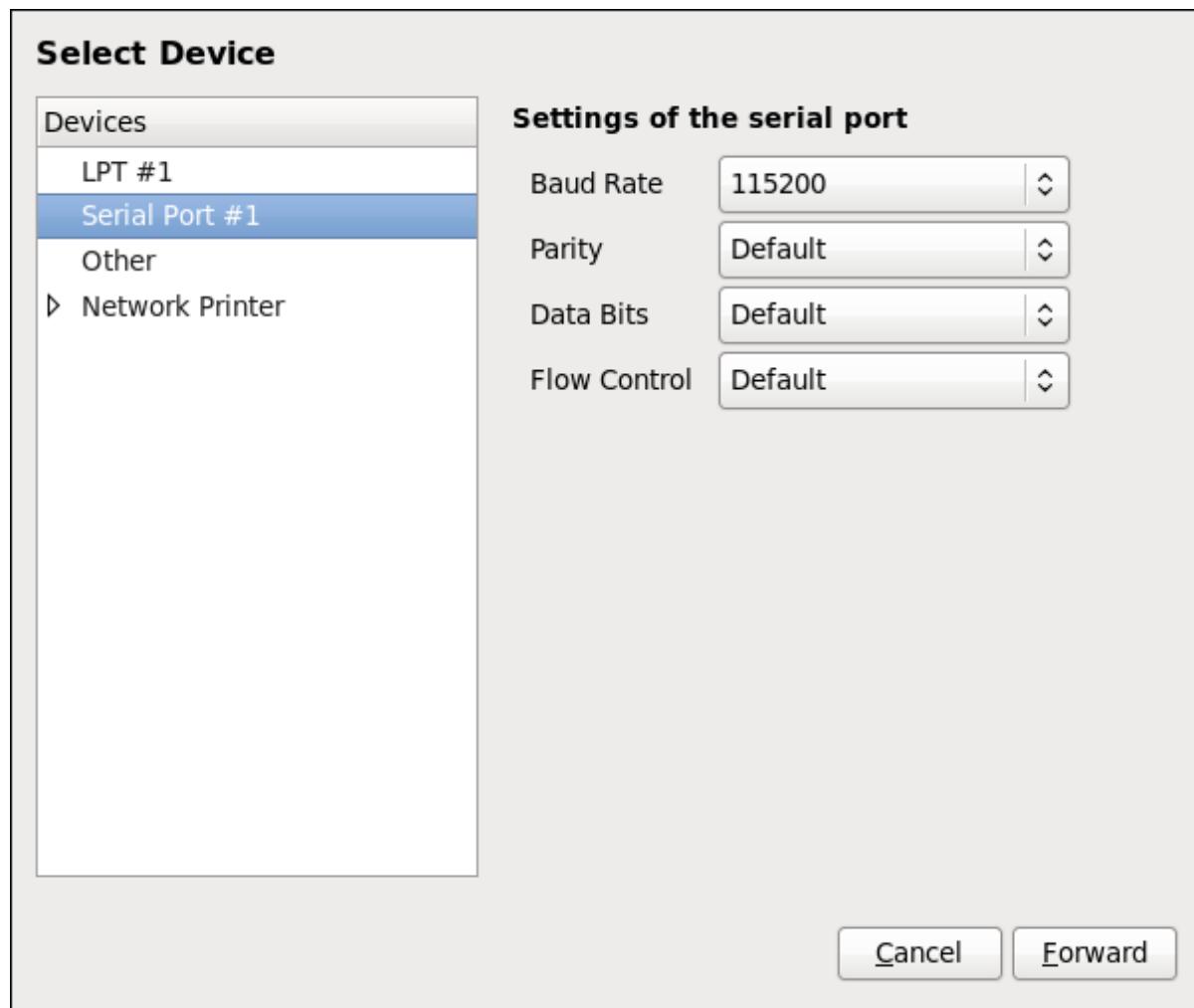


Figure 21.4. Adding a local printer

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

21.3.4. Adding an AppSocket/HP JetDirect printer

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

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

Hostname

Printer host name or IP address.

Port Number

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

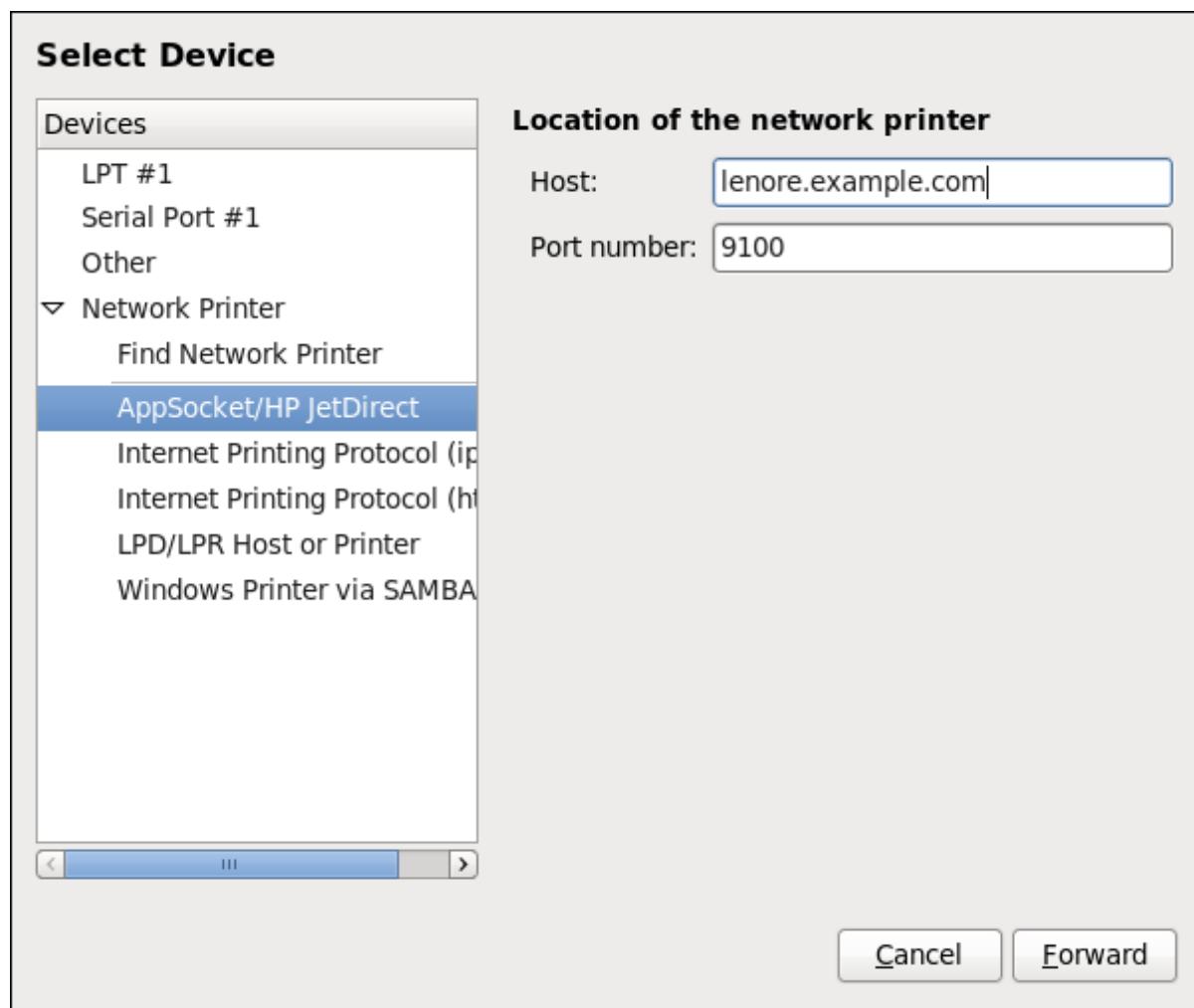


Figure 21.5. Adding a JetDirect printer

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

21.3.5. Adding an IPP Printer

An IPP printer is a printer attached to a different system on the same TCP/IP network. The system this printer is attached to may either be running CUPS or configured to use IPP.

If a firewall is enabled on the printer server, then the firewall must be configured to allow incoming TCP connections on port 631. Note that the CUPS browsing protocol allows client machines to discover shared CUPS queues automatically. To enable this, the firewall on the client machine must be configured to allow incoming UDP packets on port 631.

Follow this procedure to add an IPP printer:

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

Host

The host name of the IPP printer.

Queue

The queue name to be given to the new queue (if the box is left empty, a name based on the device node will be used).

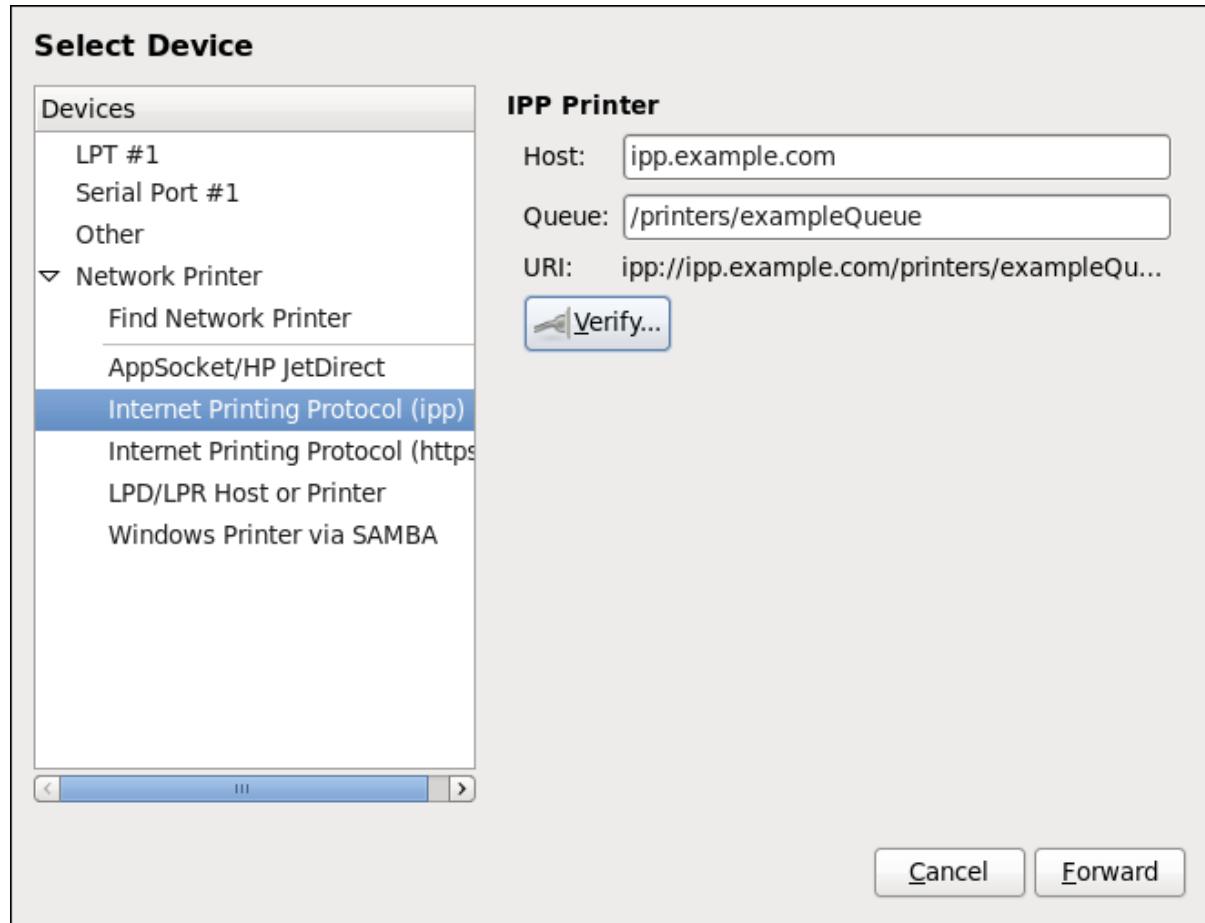


Figure 21.6. Adding an IPP printer

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

21.3.6. Adding an LPD/LPR Host or Printer

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

1. Open the **New Printer** dialog (see [Section 21.3.2, "Starting Printer Setup"](#)).
2. In the list of devices on the left, select **Network Printer → LPD/LPR Host or Printer**.
3. On the right, enter the connection settings:

Host

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

Optionally, click **Probe** to find queues on the LPD host.

Queue

The queue name to be given to the new queue (if the box is left empty, a name based on the device node will be used).

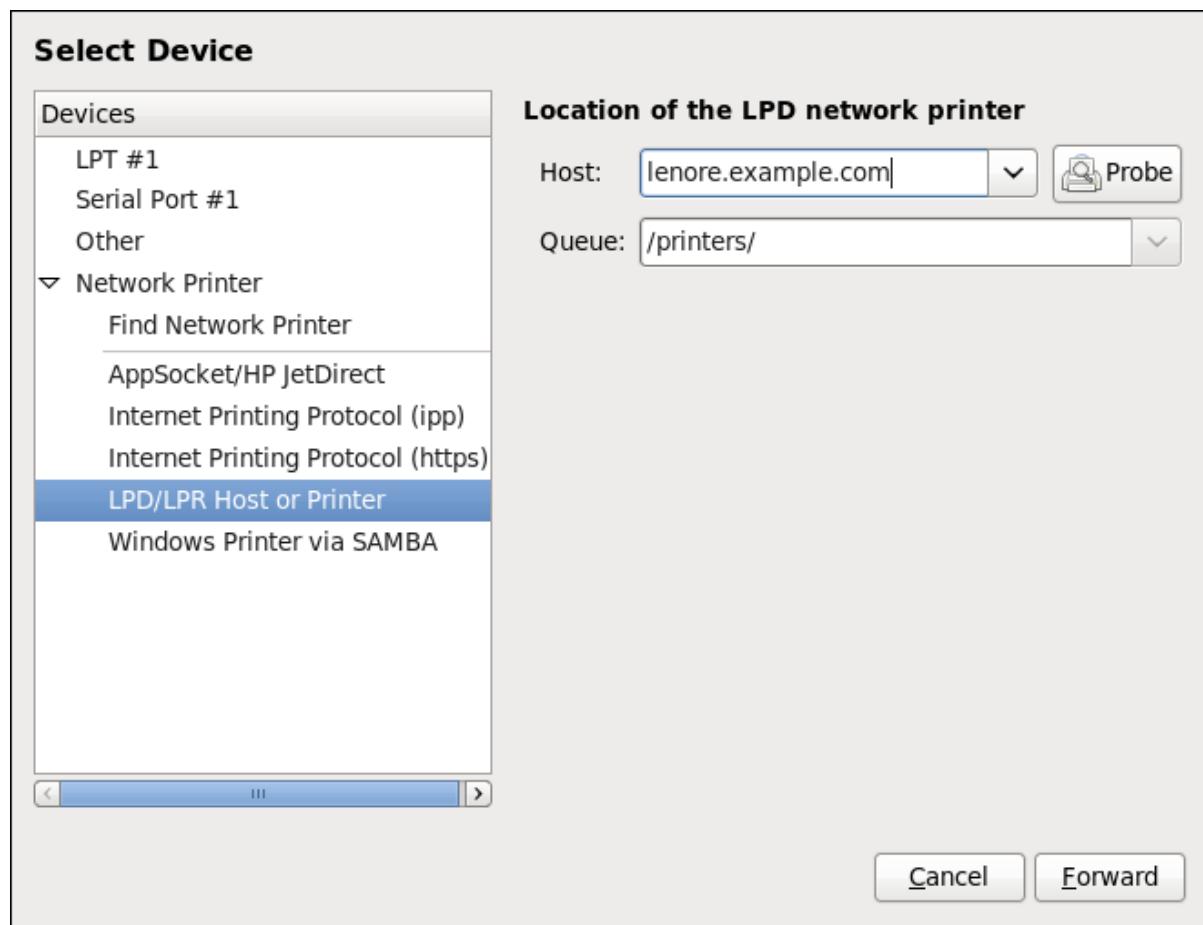


Figure 21.7. Adding an LPD/LPR printer

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

21.3.7. Adding a Samba (SMB) printer

Follow this procedure to add a Samba printer:



NOTE

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

```
yum install samba-client
```

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

1. Open the **New Printer** dialog (see [Section 21.3.2, “Starting Printer Setup”](#)).

2. In the list on the left, select **Network Printer** → **Windows Printer via SAMBA**
3. Enter the SMB address in the **smb://** field. Use the format *computer name/printer share*. In Figure 21.8, "Adding a SMB printer", the computer name is **dellbox** and the printer share is **r2**.

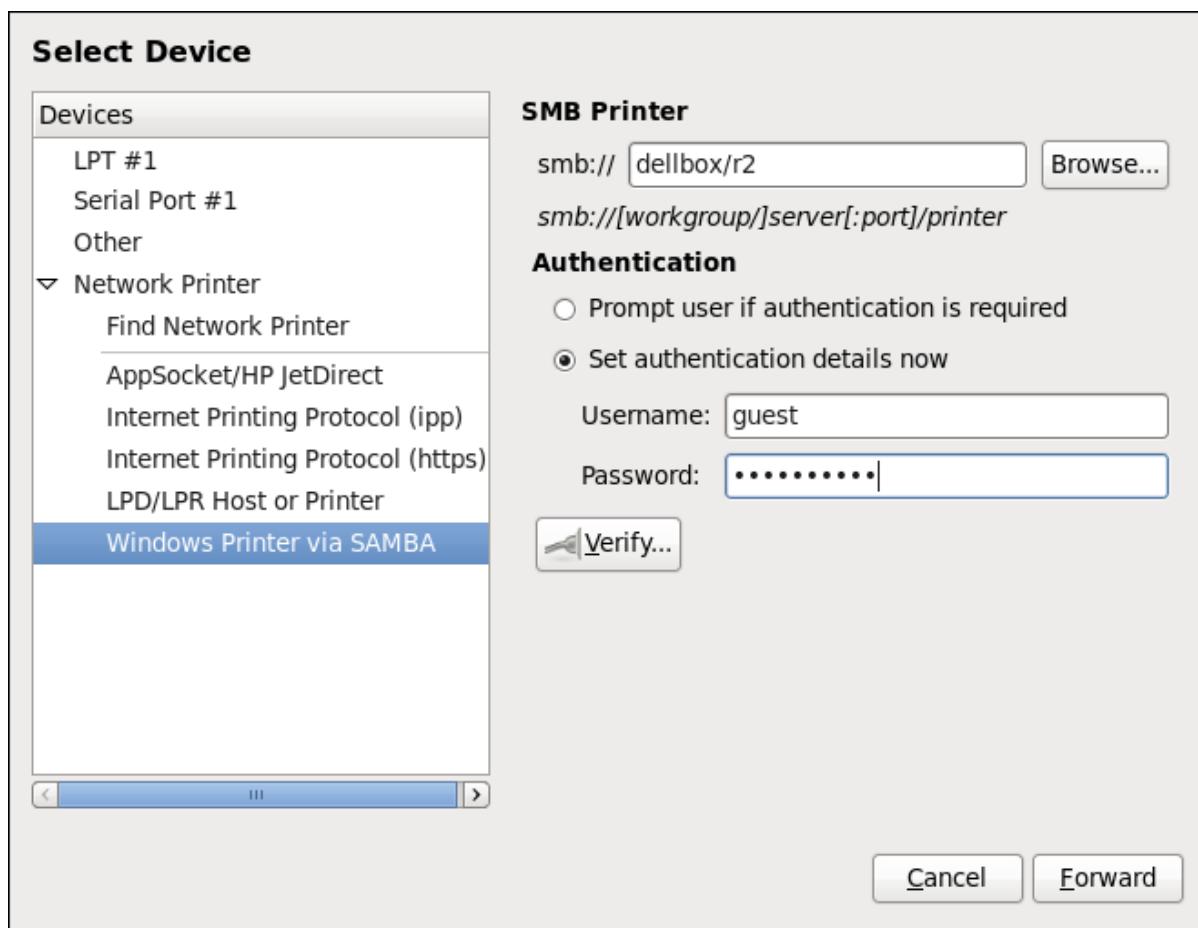


Figure 21.8. Adding a SMB printer

4. Click **Browse** to see the available workgroups/domains. To display only queues of a particular host, type in the host name (NetBios name) and click **Browse**.
5. Select either of the options:
 - **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.



WARNING

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

Therefore, when you choose a user name and password to access a Samba printer, it is advisable that you choose a password that is different from what you use to access your local Red Hat Enterprise Linux system.

If there are files shared on the Samba print server, it is recommended that they also use a password different from what is used by the print queue.

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

21.3.8. Selecting the Printer Model and Finishing

Once you have properly selected a printer connection type, the system attempts to acquire a driver. If the process fails, you can locate or search for the driver resources manually.

Follow this procedure to provide the printer driver and finish the installation:

1. In the window displayed after the automatic driver detection has failed, select one of the following options:
 - **Select a Printer from database** – the system chooses a driver based on the selected make of your printer from the list of **Makes**. If your printer model is not listed, choose **Generic**.
 - **Provide PPD file** – the system uses the provided PostScript Printer Description (PPD) file for installation. A PPD file may also be delivered with your printer as being normally provided by the manufacturer. If the PPD file is available, you can choose this option and use the browser bar below the option description to select the PPD file.
 - **Search for a printer driver to download** – enter the make and model of your printer into the **Make and model** field to search on OpenPrinting.org for the appropriate packages.

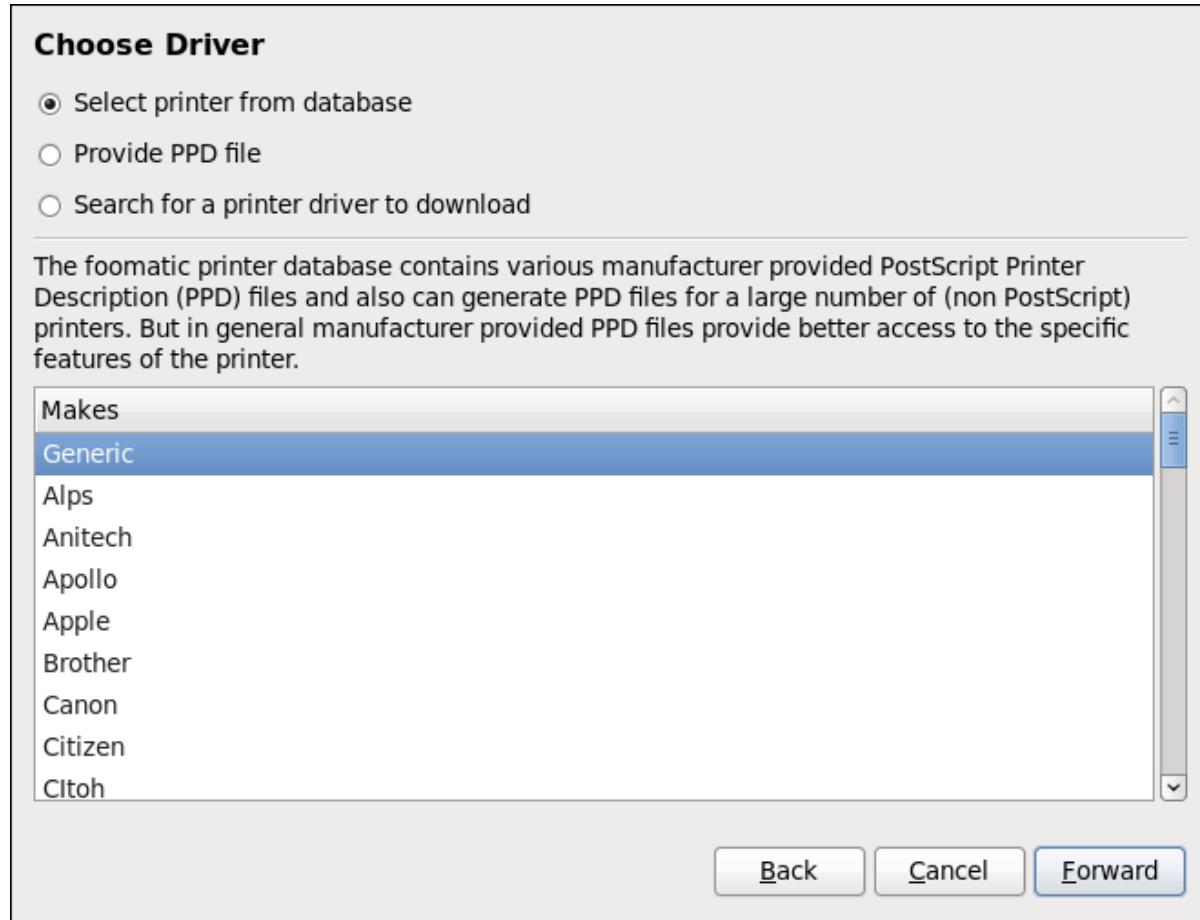


Figure 21.9. Selecting a printer brand

2. Depending on your previous choice provide details in the area displayed below:
 - Printer brand for the **Select printer from database** option.
 - PPD file location for the **Provide PPD file** option.
 - Printer make and model for the **Search for a printer driver to download** option.
3. Click **Forward** to continue.
4. If applicable for your option, window shown in [Figure 21.10, "Selecting a printer model"](#) appears. Choose the corresponding model in the **Models** column on the left.



NOTE

On the right, the recommended printer driver is automatically selected; however, you can select another available driver. The print driver processes the data that you want to print into a format the printer can understand. Since a local printer is attached directly to your computer, you need a printer driver to process the data that is sent to the printer.

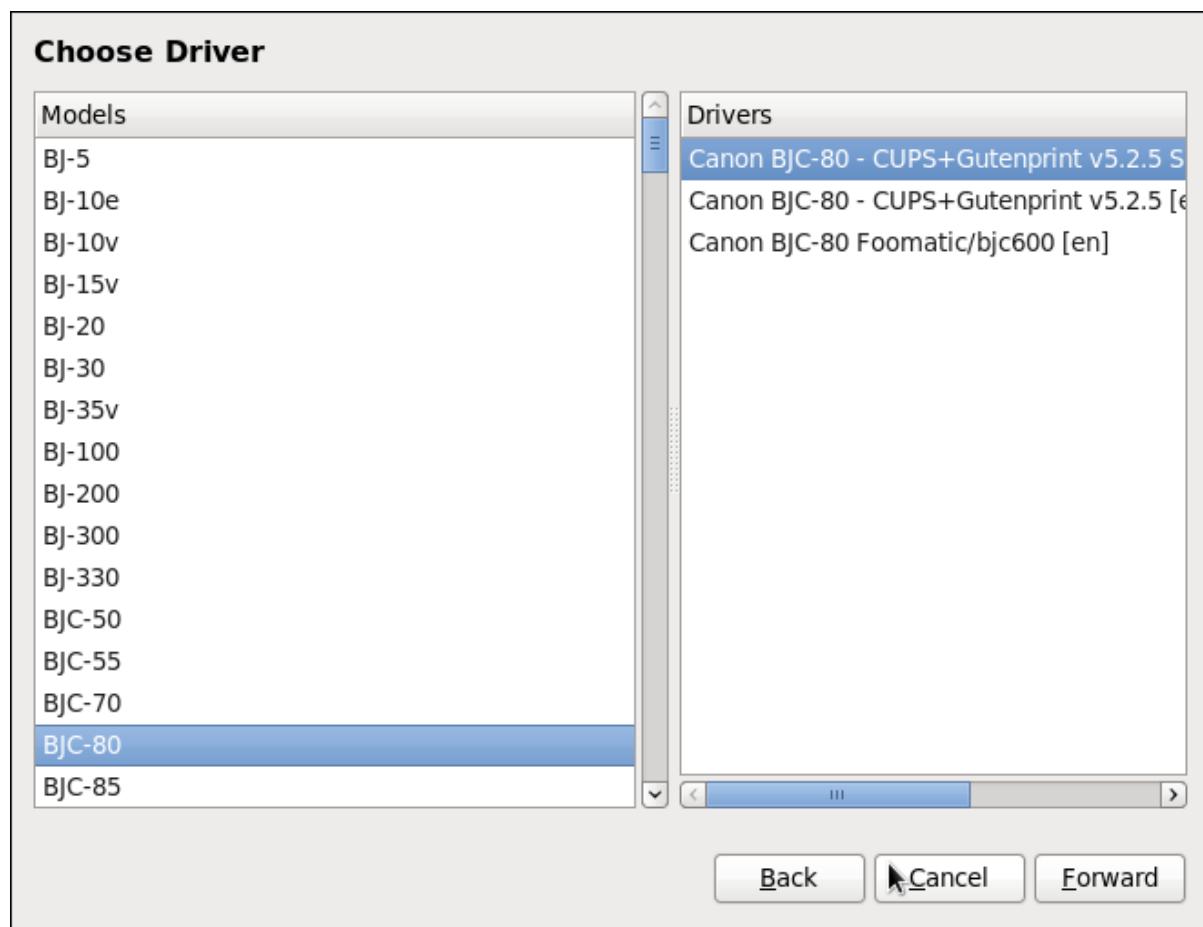


Figure 21.10. Selecting a printer model

5. Click **Forward**.
6. Under the **Describe Printer** enter a unique name for the printer in the **Printer Name** field. The printer name can contain letters, numbers, dashes (-), and underscores (_); it *must not* contain any spaces. You can also use the **Description** and **Location** fields to add further printer information. Both fields are optional, and may contain spaces.

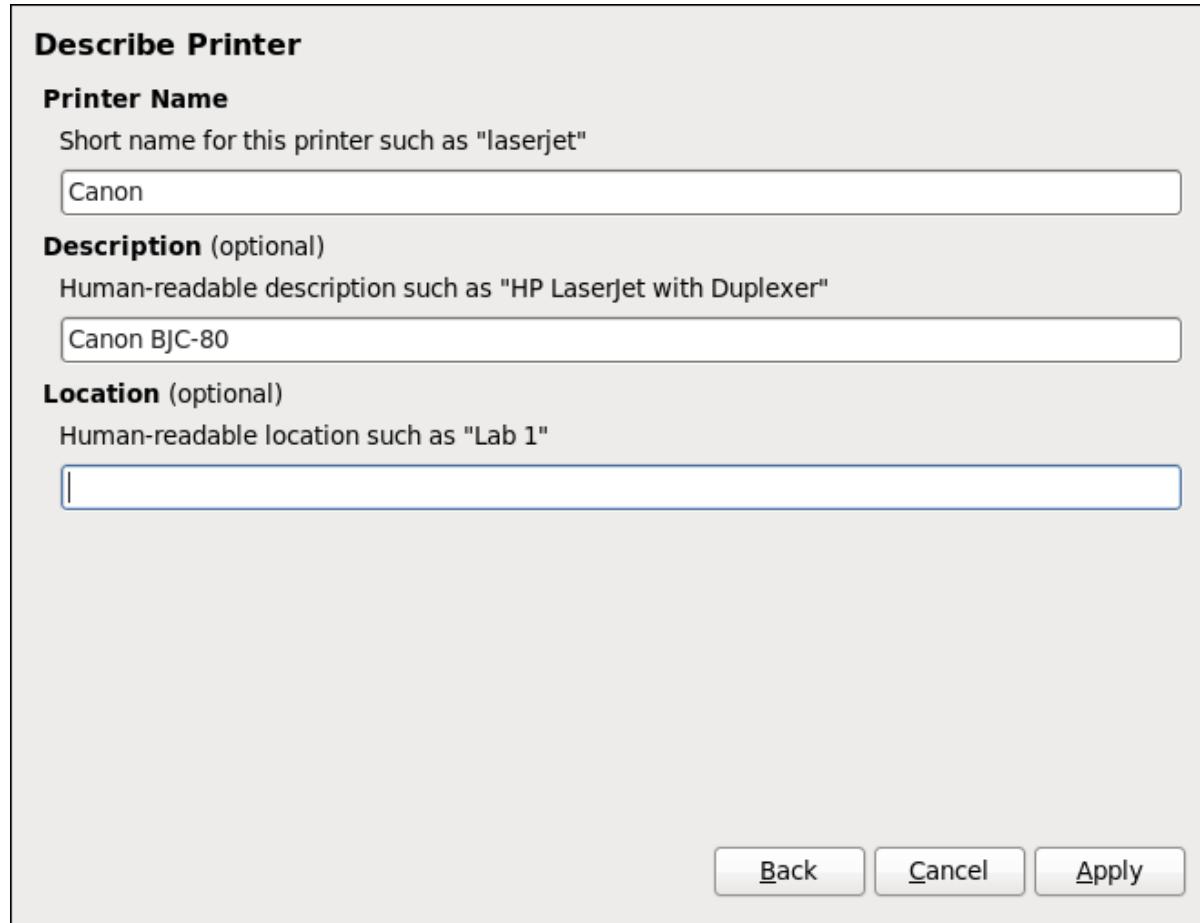


Figure 21.11. Printer setup

7. Click **Apply** to confirm your printer configuration and add the print queue if the settings are correct. Click **Back** to modify the printer configuration.
8. After the changes are applied, a dialog box appears allowing you to print a test page. Click **Yes** to print a test page now. Alternatively, you can print a test page later as described in [Section 21.3.9, “Printing a Test Page”](#).

21.3.9. Printing a Test Page

After you have set up a printer or changed a printer configuration, print a test page to make sure the printer is functioning properly:

1. Right-click the printer in the **Printing** window and click **Properties**.
2. In the Properties window, click **Settings** on the left.
3. On the displayed **Settings** tab, click the **Print Test Page** button.

21.3.10. Modifying Existing Printers

To delete an existing printer, in the **Printer Configuration** window, select the printer and go to **Printer → Delete**. Confirm the printer deletion. Alternatively, press the **Delete** key.

To set the default printer, right-click the printer in the printer list and click the **Set as Default** button in the context menu.

21.3.10.1. The Settings Page

To change printer driver configuration, double-click the corresponding name in the **Printer** list and click the **Settings** label on the left to display the **Settings** page.

You can modify printer settings such as make and model, print a test page, change the device location (URI), and more.

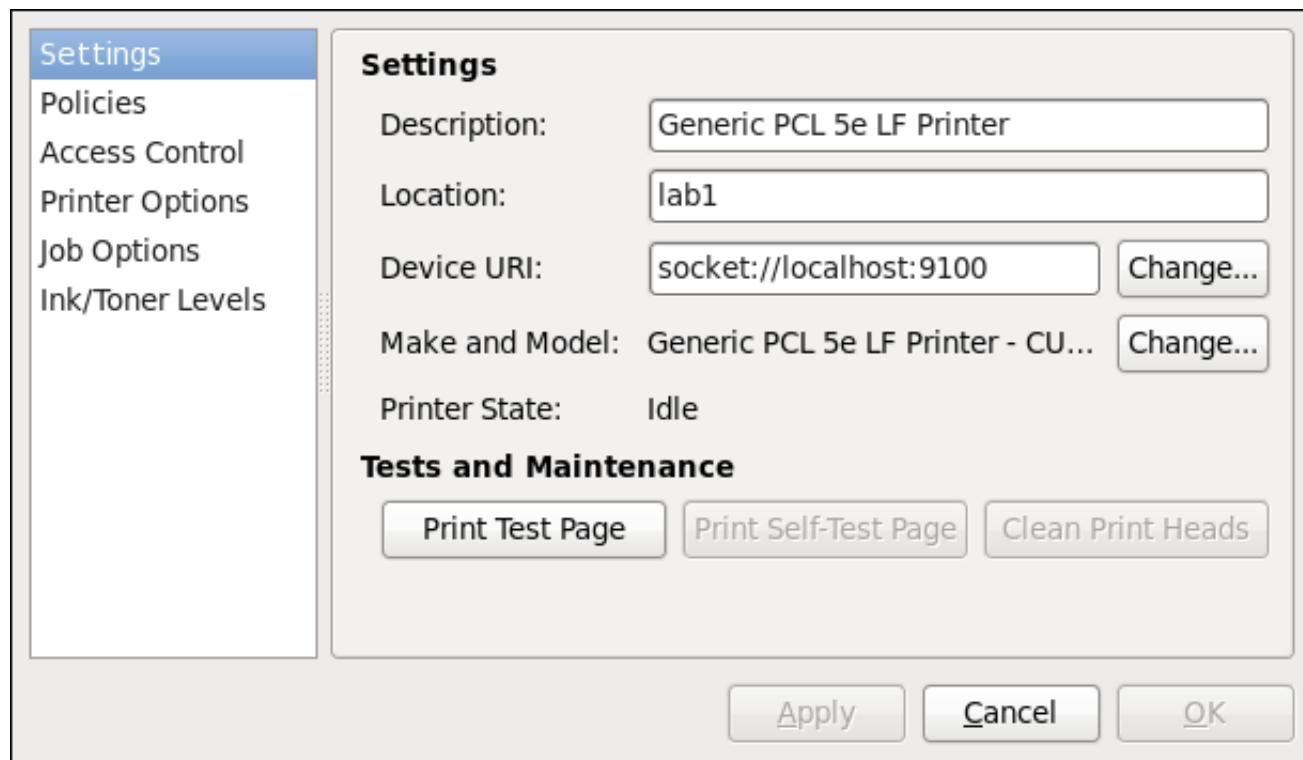


Figure 21.12. Settings page

21.3.10.2. The Policies Page

Click the **Policies** button on the left to change settings in printer state and print output.

You can select the printer states, configure the **Error Policy** of the printer (you can decide to abort the print job, retry, or stop it if an error occurs).

You can also create a *banner page* (a page that describes aspects of the print job such as the originating printer, the user name from which the job originated, and the security status of the document being printed): click the **Starting Banner** or **Ending Banner** drop-down menu and choose the option that best describes the nature of the print jobs (for example, **confidential**).

21.3.10.2.1. Sharing Printers

On the **Policies** page, you can mark a printer as shared: if a printer is shared, users published on the network can use it. To allow the sharing function for printers, go to **Server → Settings** and select **Publish shared printers connected to this system**.

Finally, make sure that the firewall allows incoming TCP connections to port 631 (that is Network Printing Server (IPP) in system-config-firewall).

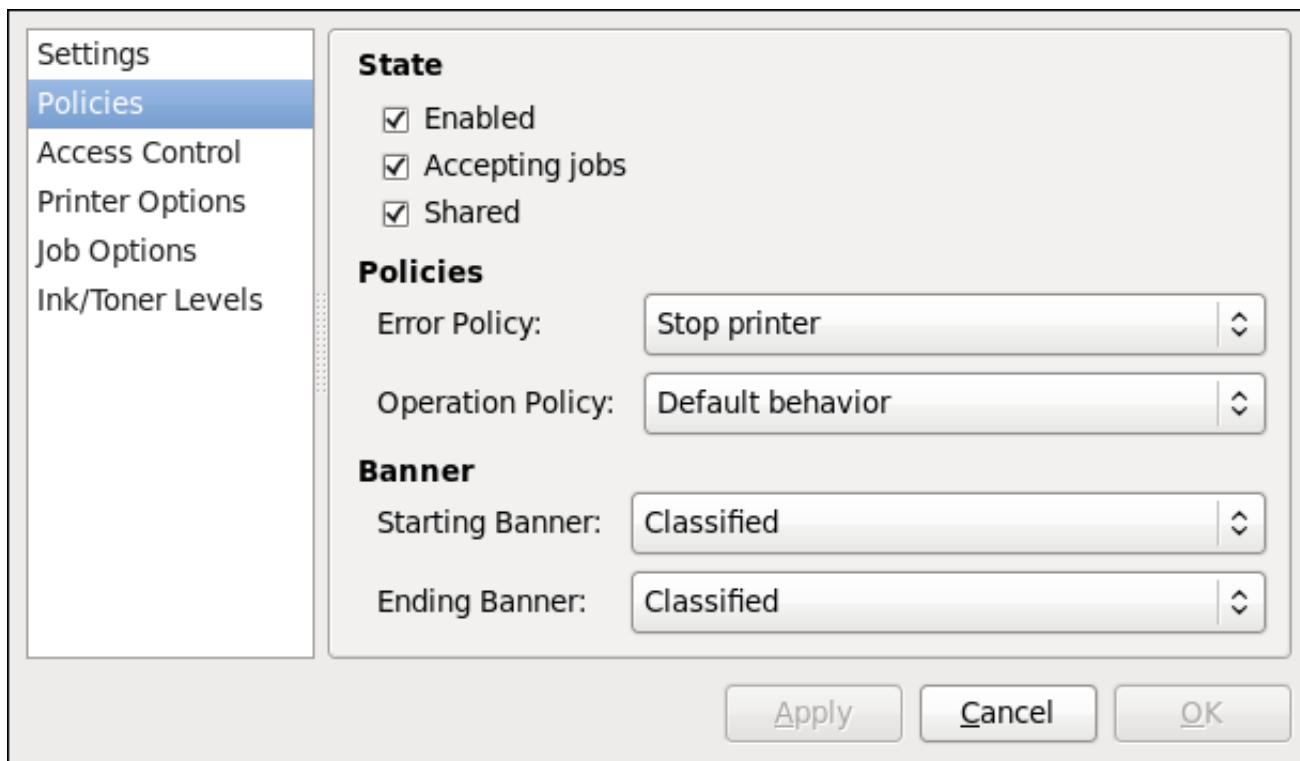


Figure 21.13. Policies page

21.3.10.2.2. The Access Control Page

You can change user-level access to the configured printer on the **Access Control** page. Click the **Access Control** label on the left to display the page. Select either **Allow printing for everyone except these users** or **Deny printing for everyone except these users** and define the user set below: enter the user name in the text box and click the **Add** button to add the user to the user set.

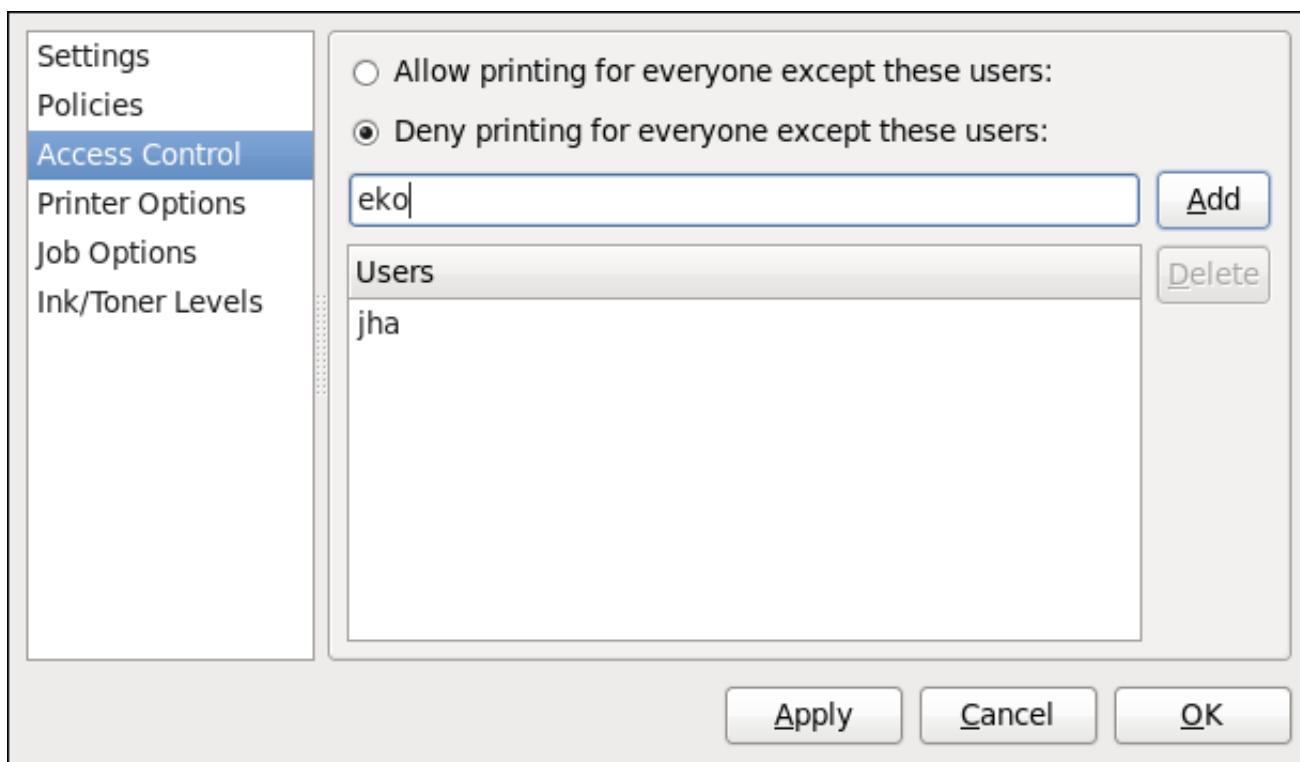


Figure 21.14. Access Control page

21.3.10.2.3. The Printer Options Page

The **Printer Options** page contains various configuration options for the printer media and output, and its content may vary from printer to printer. It contains general printing, paper, quality, and printing size settings.

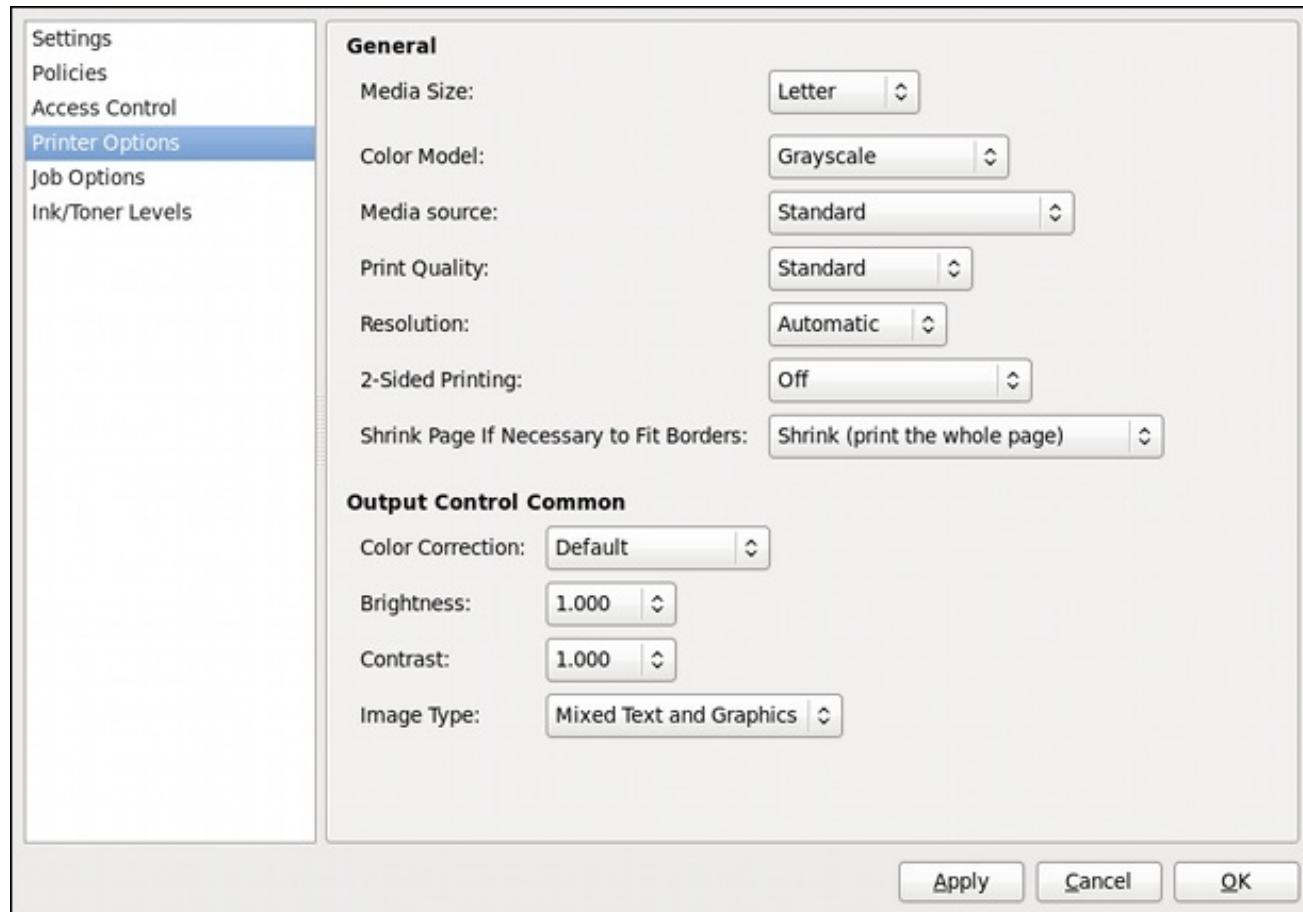


Figure 21.15. Printer Options page

21.3.10.2.4. Job Options Page

On the **Job Options** page, you can detail the printer job options. Click the **Job Options** label on the left to display the page. Edit the default settings to apply custom job options, such as number of copies, orientation, pages per side, scaling (increase or decrease the size of the printable area, which can be used to fit an oversize print area onto a smaller physical sheet of print medium), detailed text options, and custom job options.

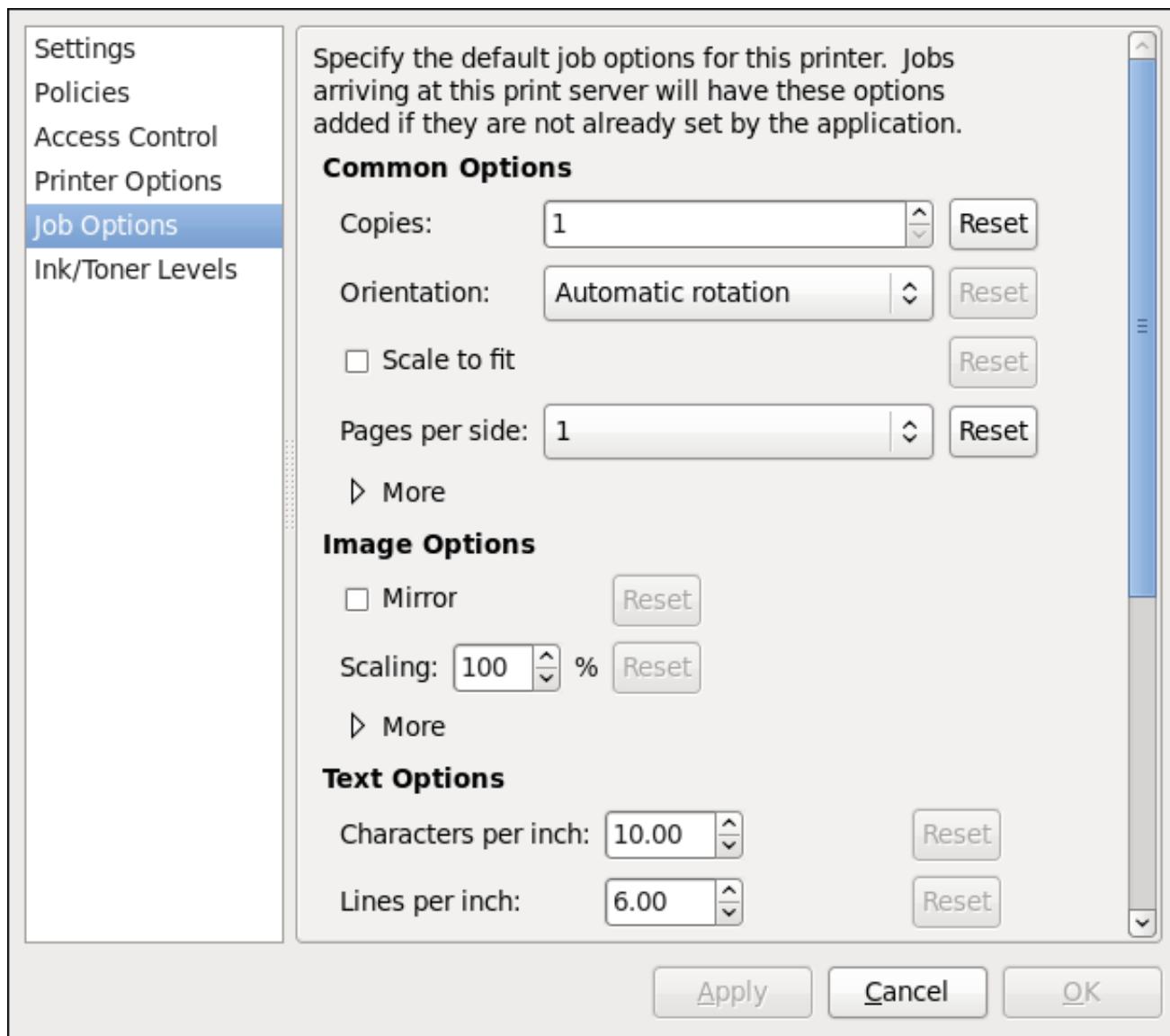


Figure 21.16. Job Options page

21.3.10.2.5. Ink/Toner Levels Page

The **Ink/Toner Levels** page contains details on toner status if available and printer status messages. Click the **Ink/Toner Levels** label on the left to display the page.

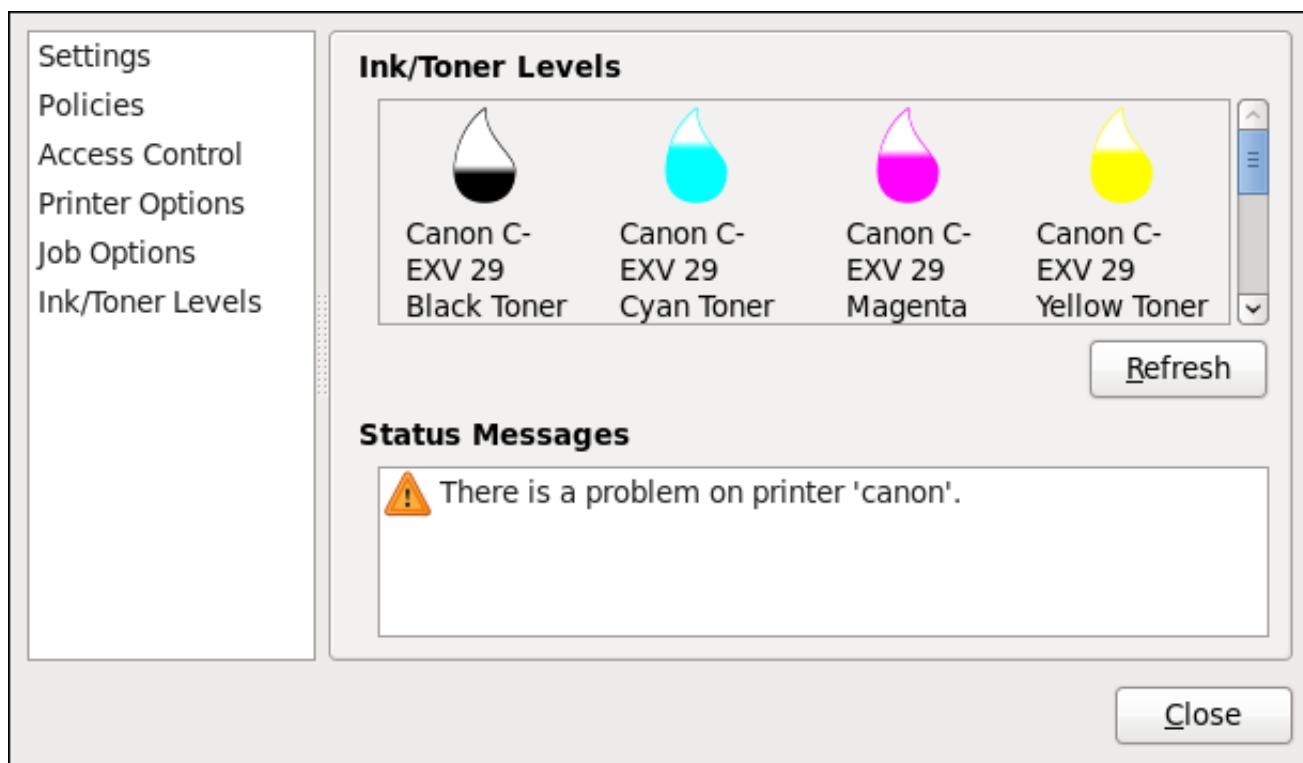


Figure 21.17. Ink/Toner Levels page

21.3.10.3. Managing Print Jobs

When you send a print job to the printer daemon, such as printing a text file from **Emacs** or printing an image from **GIMP**, the print job is added to the print spool queue. The print spool queue is a list of print jobs that have been sent to the printer and information about each print request, such as the status of the request, the job number, and more.

During the printing process, the **Printer Status** icon appears in the **Notification Area** on the panel. To check the status of a print job, click the **Printer Status**, which displays a window similar to [Figure 21.18, "GNOME Print Status"](#).

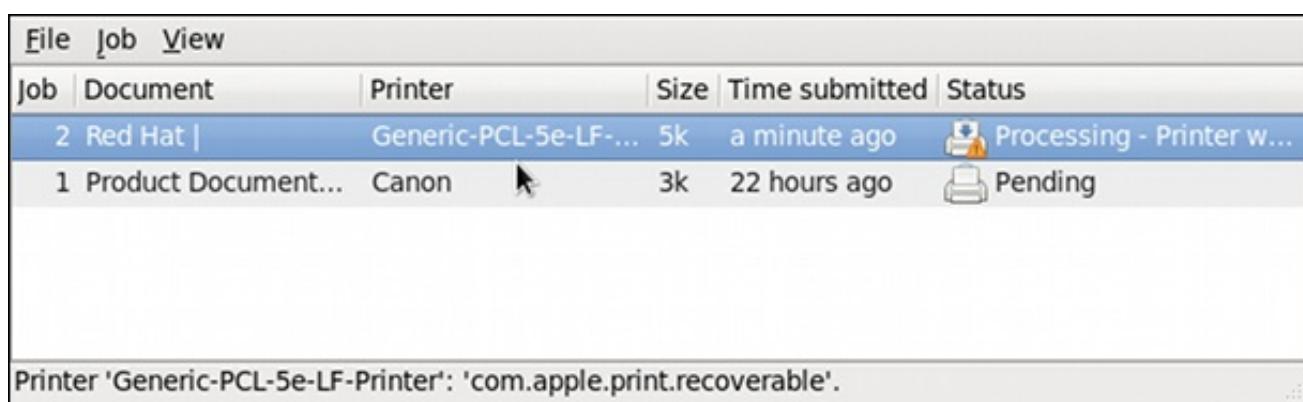


Figure 21.18. GNOME Print Status

To cancel, hold, release, reprint or authenticate a print job, select the job in the **GNOME Print Status** and on the **Job** menu, click the respective command.

To view the list of print jobs in the print spool from a shell prompt, type the command **lpstat -o**. The last few lines look similar to the following:

Example 21.11. Example of **lpstat -o output**

```
$ lpstat -o
Charlie-60      twaugh      1024  Tue 08 Feb 2011 16:42:11 GMT
Aaron-61        twaugh      1024  Tue 08 Feb 2011 16:42:44 GMT
Ben-62          root        1024  Tue 08 Feb 2011 16:45:42 GMT
```

If you want to cancel a print job, find the job number of the request with the command **lpstat -o** and then use the command **cancel job number**. For example, **cancel 60** would cancel the print job in [Example 21.11, “Example of **lpstat -o** output”](#). You can not cancel print jobs that were started by other users with the **cancel** command. However, you can enforce deletion of such job by issuing the **cancel -U root job_number** command. To prevent such canceling change the printer operation policy to **Authenticated** to force root authentication.

You can also print a file directly from a shell prompt. For example, the command **lp sample.txt** prints the text file **sample.txt**. The print filter determines what type of file it is and converts it into a format the printer can understand.

21.3.11. Additional Resources

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

21.3.11.1. Installed Documentation

man lp

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

man lpr

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

man cancel

The manual page for the command-line utility to remove print jobs from the print queue.

man mpage

The manual page for the command-line utility to print multiple pages on one sheet of paper.

man cupsd

The manual page for the CUPS printer daemon.

man cupsd.conf

The manual page for the CUPS printer daemon configuration file.

man classes.conf

The manual page for the class configuration file for CUPS.

man lpstat

The manual page for the **lpstat** command, which displays status information about classes, jobs, and printers.

21.3.11.2. Useful Websites

<http://www.linuxprinting.org/>

GNU/Linux Printing contains a large amount of information about printing in Linux.

<http://www.cups.org/>

Documentation, FAQs, and newsgroups about CUPS.

CHAPTER 22. CONFIGURING NTP USING NTPD

22.1. INTRODUCTION TO NTP

The *Network Time Protocol* (NTP) enables the accurate dissemination of time and date information in order to keep the time clocks on networked computer systems synchronized to a common reference over the network or the Internet. Many standards bodies around the world have atomic clocks which may be made available as a reference. The satellites that make up the Global Position System contain more than one atomic clock, making their time signals potentially very accurate. Their signals can be deliberately degraded for military reasons. An ideal situation would be where each site has a server, with its own reference clock attached, to act as a site-wide time server. Many devices which obtain the time and date via low frequency radio transmissions or the Global Position System (GPS) exist. However for most situations, a range of publicly accessible time servers connected to the Internet at geographically dispersed locations can be used. These **NTP** servers provide “*Coordinated Universal Time*” (UTC). Information about these time servers can be found at www.pool.ntp.org.

Accurate time keeping is important for a number of reasons in IT. In networking for example, accurate time stamps in packets and logs are required. Logs are used to investigate service and security issues and so timestamps made on different systems must be made by synchronized clocks to be of real value. As systems and networks become increasingly faster, there is a corresponding need for clocks with greater accuracy and resolution. In some countries there are legal obligations to keep accurately synchronized clocks. Please see www.ntp.org for more information. In Linux systems, **NTP** is implemented by a daemon running in user space. The default **NTP** daemon in Red Hat Enterprise Linux 6 is **ntpd**.

The user space daemon updates the system clock, which is a software clock running in the kernel. Linux uses a software clock as its system clock for better resolution than the typical embedded hardware clock referred to as the “*Real Time Clock*” (RTC). See the **rtc(4)** and **hwclock(8)** man pages for information on hardware clocks. The system clock can keep time by using various clock sources. Usually, the *Time Stamp Counter* (TSC) is used. The TSC is a CPU register which counts the number of cycles since it was last reset. It is very fast, has a high resolution, and there are no interrupts. On system start, the system clock reads the time and date from the RTC. The time kept by the RTC will drift away from actual time by up to 5 minutes per month due to temperature variations. Hence the need for the system clock to be constantly synchronized with external time references. When the system clock is being synchronized by **ntpd**, the kernel will in turn update the RTC every 11 minutes automatically.

22.2. NTP STRATA

NTP servers are classified according to their synchronization distance from the atomic clocks which are the source of the time signals. The servers are thought of as being arranged in layers, or strata, from 1 at the top down to 15. Hence the word stratum is used when referring to a specific layer. Atomic clocks are referred to as Stratum 0 as this is the source, but no Stratum 0 packet is sent on the Internet, all stratum 0 atomic clocks are attached to a server which is referred to as stratum 1. These servers send out packets marked as Stratum 1. A server which is synchronized by means of packets marked stratum **n** belongs to the next, lower, stratum and will mark its packets as stratum **n+1**. Servers of the same stratum can exchange packets with each other but are still designated as belonging to just the one stratum, the stratum one below the best reference they are synchronized to. The designation Stratum 16 is used to indicate that the server is not currently synchronized to a reliable time source.

Note that by default **NTP** clients act as servers for those systems in the stratum below them.

Here is a summary of the **NTP** Strata:

Stratum 0:

Atomic Clocks and their signals broadcast over Radio and GPS

- GPS (Global Positioning System)
- Mobile Phone Systems
- Low Frequency Radio Broadcasts WWVB (Colorado, USA.), JJY-40 and JJY-60 (Japan), DCF77 (Germany), and MSF (United Kingdom)

These signals can be received by dedicated devices and are usually connected by RS-232 to a system used as an organizational or site-wide time server.

Stratum 1:

Computer with radio clock, GPS clock, or atomic clock attached

Stratum 2:

Reads from stratum 1; Serves to lower strata

Stratum 3:

Reads from stratum 2; Serves to lower strata

Stratum $n+1$:

Reads from stratum n ; Serves to lower strata

Stratum 15:

Reads from stratum 14; This is the lowest stratum.

This process continues down to Stratum 15 which is the lowest valid stratum. The label Stratum 16 is used to indicate an unsynchronized state.

22.3. UNDERSTANDING NTP

The version of **NTP** used by Red Hat Enterprise Linux is as described in [RFC 1305 Network Time Protocol \(Version 3\) Specification, Implementation and Analysis](#) and [RFC 5905 Network Time Protocol Version 4: Protocol and Algorithms Specification](#)

This implementation of **NTP** enables sub-second accuracy to be achieved. Over the Internet, accuracy to 10s of milliseconds is normal. On a Local Area Network (LAN), 1ms accuracy is possible under ideal conditions. This is because clock drift is now accounted and corrected for, which was not done in earlier, simpler, time protocol systems. A resolution of 233 picoseconds is provided by using 64-bit time stamps. The first 32-bits of the time stamp is used for seconds, the last 32-bits are used for fractions of seconds.

NTP represents the time as a count of the number of seconds since 00:00 (midnight) 1 January, 1900 GMT. As 32-bits is used to count the seconds, this means the time will "roll over" in 2036. However **NTP** works on the difference between time stamps so this does not present the same level of problem as other implementations of time protocols have done. If a hardware clock that is within 68 years of the correct time is available at boot time then **NTP** will correctly interpret the current date. The **NTP4** specification provides for an "Era Number" and an "Era Offset" which can be used to make software more robust when dealing with time lengths of more than 68 years. Note, please do not confuse this with the Unix Year 2038 problem.

The **NTP** protocol provides additional information to improve accuracy. Four time stamps are used to

allow the calculation of round-trip time and server response time. In order for a system in its role as **NTP** client to synchronize with a reference time server, a packet is sent with an “originate time stamp”. When the packet arrives, the time server adds a “receive time stamp”. After processing the request for time and date information and just before returning the packet, it adds a “transmit time stamp”. When the returning packet arrives at the **NTP** client, a “receive time stamp” is generated. The client can now calculate the total round trip time and by subtracting the processing time derive the actual traveling time. By assuming the outgoing and return trips take equal time, the single-trip delay in receiving the **NTP** data is calculated. The full **NTP** algorithm is much more complex than presented here.

When a packet containing time information is received it is not immediately responded to, but is first subject to validation checks and then processed together with several other time samples to arrive at an estimate of the time. This is then compared to the system clock to determine the time offset, the difference between the system clock's time and what **ntpd** has determined the time should be. The system clock is adjusted slowly, at most at a rate of 0.5ms per second, to reduce this offset by changing the frequency of the counter being used. It will take at least 2000 seconds to adjust the clock by 1 second using this method. This slow change is referred to as slewing and cannot go backwards. If the time offset of the clock is more than 128ms (the default setting), **ntpd** can “step” the clock forwards or backwards. If the time offset at system start is greater than 1000 seconds then the user, or an installation script, should make a manual adjustment. See [Chapter 2, Date and Time Configuration](#). With the **-g** option to the **ntpd** command (used by default), any offset at system start will be corrected, but during normal operation only offsets of up to 1000 seconds will be corrected.

Some software may fail or produce an error if the time is changed backwards. For systems that are sensitive to step changes in the time, the threshold can be changed to 600s instead of 128ms using the **-x** option (unrelated to the **-g** option). Using the **-x** option to increase the stepping limit from 0.128s to 600s has a drawback because a different method of controlling the clock has to be used. It disables the kernel clock discipline and may have a negative impact on the clock accuracy. The **-x** option can be added to the `/etc/sysconfig/ntpd` configuration file.

22.4. UNDERSTANDING THE DRIFT FILE

The drift file is used to store the frequency offset between the system clock running at its nominal frequency and the frequency required to remain in synchronization with UTC. If present, the value contained in the drift file is read at system start and used to correct the clock source. Use of the drift file reduces the time required to achieve a stable and accurate time. The value is calculated, and the drift file replaced, once per hour by **ntpd**. The drift file is replaced, rather than just updated, and for this reason the drift file must be in a directory for which **ntpd** has write permissions.

22.5. UTC, TIMEZONES, AND DST

As **NTP** is entirely in UTC (Universal Time, Coordinated), Timezones and DST (Daylight Saving Time) are applied locally by the system. The file `/etc/localtime` is a copy of, or symlink to, a zone information file from `/usr/share/zoneinfo`. The RTC may be in localtime or in UTC, as specified by the 3rd line of `/etc/adjtime`, which will be one of LOCAL or UTC to indicate how the RTC clock has been set. Users can easily change this setting using the check box **System Clock Uses UTC** in the `system-config-date` graphical configuration tool. See [Chapter 2, Date and Time Configuration](#) for information on how to use that tool. Running the RTC in UTC is recommended to avoid various problems when daylight saving time is changed.

The operation of **ntpd** is explained in more detail in the man page [**ntpd\(8\)**](#). The resources section lists useful sources of information. See [Section 22.19, “Additional Resources”](#).

22.6. AUTHENTICATION OPTIONS FOR NTP

NTPv4 added support for the Autokey Security Architecture, which is based on public asymmetric

cryptography while retaining support for symmetric key cryptography. The Autokey Security Architecture is described in [RFC 5906 Network Time Protocol Version 4: Autokey Specification](#). The man page **ntp_auth(5)** describes the authentication options and commands for **ntpd**.

An attacker on the network can attempt to disrupt a service by sending **NTP** packets with incorrect time information. On systems using the public pool of **NTP** servers, this risk is mitigated by having more than three **NTP** servers in the list of public **NTP** servers in **/etc/ntp.conf**. If only one time source is compromised or spoofed, **ntpd** will ignore that source. You should conduct a risk assessment and consider the impact of incorrect time on your applications and organization. If you have internal time sources you should consider steps to protect the network over which the **NTP** packets are distributed. If you conduct a risk assessment and conclude that the risk is acceptable, and the impact to your applications minimal, then you can choose not to use authentication.

The broadcast and multicast modes require authentication by default. If you have decided to trust the network then you can disable authentication by using **disable auth** directive in the **ntp.conf** file. Alternatively, authentication needs to be configured by using SHA1 or MD5 symmetric keys, or by public (asymmetric) key cryptography using the Autokey scheme. The Autokey scheme for asymmetric cryptography is explained in the **ntp_auth(8)** man page and the generation of keys is explained in **ntp-keygen(8)**. To implement symmetric key cryptography, see [Section 22.16.12, “Configuring Symmetric Authentication Using a Key”](#) for an explanation of the **key** option.

22.7. MANAGING THE TIME ON VIRTUAL MACHINES

Virtual machines cannot access a real hardware clock and a virtual clock is not stable enough as the stability is dependent on the host systems work load. For this reason, para-virtualized clocks should be provided by the virtualization application in use. On Red Hat Enterprise Linux with **KVM** the default clock source is **kvm-clock**. See the [KVM guest timing management](#) chapter of the *Virtualization Host Configuration and Guest Installation Guide*.

22.8. UNDERSTANDING LEAP SECONDS

Greenwich Mean Time (GMT) was derived by measuring the solar day, which is dependent on the Earth's rotation. When atomic clocks were first made, the potential for more accurate definitions of time became possible. In 1958, International Atomic Time (TAI) was introduced based on the more accurate and very stable atomic clocks. A more accurate astronomical time, Universal Time 1 (UT1), was also introduced to replace GMT. The atomic clocks are in fact far more stable than the rotation of the Earth and so the two times began to drift apart. For this reason UTC was introduced as a practical measure. It is kept within one second of UT1 but to avoid making many small trivial adjustments it was decided to introduce the concept of a *leap second* in order to reconcile the difference in a manageable way. The difference between UT1 and UTC is monitored until they drift apart by more than half a second. Then only is it deemed necessary to introduce a one second adjustment, forward or backward. Due to the erratic nature of the Earth's rotational speed, the need for an adjustment cannot be predicted far into the future. The decision as to when to make an adjustment is made by the [International Earth Rotation and Reference Systems Service \(IERS\)](#). However, these announcements are important only to administrators of Stratum 1 servers because **NTP** transmits information about pending leap seconds and applies them automatically.

22.9. UNDERSTANDING THE NTPD CONFIGURATION FILE

The daemon, **ntpd**, reads the configuration file at system start or when the service is restarted. The default location for the file is **/etc/ntp.conf** and you can view the file by entering the following command:

```
~]$ less /etc/ntp.conf
```

The configuration commands are explained briefly later in this chapter, see [Section 22.16, "Configure NTP"](#), and more verbosely in the **ntp.conf(5)** man page.

Here follows a brief explanation of the contents of the default configuration file:

The driftfile entry

A path to the drift file is specified, the default entry on Red Hat Enterprise Linux is:

```
driftfile /var/lib/ntp/drift
```

If you change this be certain that the directory is writable by **ntpd**. The file contains one value used to adjust the system clock frequency after every system or service start. See [Understanding the Drift File](#) for more information.

The access control entries

The following lines setup the default access control restrictions:

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

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

Addresses within the range **127.0.0.0/8** are sometimes required by various processes or applications. As the "restrict default" line above prevents access to everything not explicitly allowed, access to the standard loopback address for **IPv4** and **IPv6** is permitted by means of the following lines:

```
# the administrative functions.
restrict 127.0.0.1
restrict -6 ::1
```

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

Hosts on the local network are not permitted because of the "restrict default" line above. To change this, for example to allow hosts from the **192.0.2.0/24** network to query the time and statistics but nothing more, a line in the following format is required:

```
restrict 192.0.2.0 mask 255.255.255.0 nomodify notrap nopeer
```

To allow unrestricted access from a specific host, for example **192.0.2.250/32**, a line in the following format is required:

```
restrict 192.0.2.250
```

A mask of **255.255.255.255** is applied if none is specified.

The restrict commands are explained in the **ntp_acc(5)** man page.

The public servers entry

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

```
server 0.rhel.pool.ntp.org iburst
server 1.rhel.pool.ntp.org iburst
server 2.rhel.pool.ntp.org iburst
server 3.rhel.pool.ntp.org iburst
```

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

The broadcast/multicast servers entry

By default, the **ntp.conf** file contains some commented out examples. These are largely self explanatory. See the explanation of the specific commands [Section 22.16, “Configure NTP”](#). If required, add your commands just below the examples.



NOTE

When the **DHCP** client program, **dhclient**, receives a list of **NTP** servers from the **DHCP** server, it adds them to **ntp.conf** and restarts the service. To disable that feature, add **PEERNTP=no** to **/etc/sysconfig/network**.

22.10. UNDERSTANDING THE NTPD SYSCONFIG FILE

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

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

The **-g** option enables **ntpd** to ignore the offset limit of 1000s and attempt to synchronize the time even if the offset is larger than 1000s, but only on system start. Without that option **ntpd** will exit if the time offset is greater than 1000s. It will also exit after system start if the service is restarted and the offset is greater than 1000s even with the **-g** option.

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

22.11. CHECKING IF THE NTP DAEMON IS INSTALLED

To check if **ntpd** is installed, enter the following command as root:

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

NTP is implemented by means of the daemon or service **ntpd**, which is contained within the **ntp** package.

22.12. INSTALLING THE NTP DAEMON (NTPD)

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

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

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

22.13. CHECKING THE STATUS OF NTP

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

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

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

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

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

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

- remote and refid: remote NTP server, and its NTP server
- st: stratum of server
- t: type of server (local, unicast, multicast, or broadcast)
- poll: how frequently to query server (in seconds)
- when: how long since last poll (in seconds)
- reach: octal bitmask of success or failure of last 8 queries (left-shifted); 377 = 1111111 = all recent queries were successful; 257 = 1010111 = 4 most recent were successful, 5 and 7 failed
- delay: network round trip time (in milliseconds)
- offset: difference between local clock and remote clock (in milliseconds)
- jitter: difference of successive time values from server (high jitter could be due to an unstable clock or, more likely, poor network performance)

To obtain a brief status report from **ntpd**, issue the following command:

```
~]$ ntpstat
unsynchronised
time server re-starting
polling server every 64 s
```

```
~]$ ntpstat
synchronised to NTP server (10.5.26.10) at stratum 2
time correct to within 52 ms
polling server every 1024 s
```

22.14. CONFIGURE THE FIREWALL TO ALLOW INCOMING NTP PACKETS

The **NTP** traffic consists of **UDP** packets on port **123** and needs to be permitted through network and host-based firewalls in order for **NTP** to function.

22.14.1. Configure the Firewall Using the Graphical Tool

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

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

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

22.14.2. Configure the Firewall Using the Command Line

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

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

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

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

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

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

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

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

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

```
~]# less /etc/sysconfig/system-config-firewall
# Configuration file for system-config-firewall
--enabled
--service=ssh
```

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

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

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

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

22.15. CONFIGURE NTPDATE SERVERS

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

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

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

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

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

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

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

The number of servers listed is not very important as **ntpdate** will only use this to obtain the date information once when the system is starting. If you have an internal time server then use that host name for the first line. An additional host on the second line as a backup is sensible. The selection of backup servers and whether the second host is internal or external depends on your risk assessment. For example, what is the chance of any problem affecting the first server also affecting the second server? Would connectivity to an external server be more likely to be available than connectivity to internal servers in the event of a network failure disrupting access to the first server?

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

22.16. CONFIGURE NTP

To change the default configuration of the **NTP** service, use a text editor running as **root** user to edit the **/etc/ntp.conf** file. This file is installed together with **ntpd** and is configured to use time servers from the Red Hat pool by default. The man page **ntp.conf(5)** describes the command options that can be used in the configuration file apart from the access and rate limiting commands which are explained in the **ntp_acc(5)** man page.

22.16.1. Configure Access Control to an NTP Service

To restrict or control access to the **NTP** service running on a system, make use of the **restrict** command in the **ntp.conf** file. See the commented out example:

```
# Hosts on local network are less restricted.  
#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
```

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

```
restrict address mask option
```

where *address* and *mask* specify the IP addresses to which you want to apply the restriction, and *option* is one or more of:

- **ignore** – All packets will be ignored, including **ntpq** and **ntpdc** queries.
- **kod** – a “Kiss-o'-death” packet is to be sent to reduce unwanted queries.
- **limited** – do not respond to time service requests if the packet violates the rate limit default values or those specified by the **discard** command. **ntpq** and **ntpdc** queries are not affected. For more information on the **discard** command and the default values, see [Section 22.16.2, “Configure Rate Limiting Access to an NTP Service”](#).
- **lowpriotrap** – traps set by matching hosts to be low priority.
- **nomodify** – prevents any changes to the configuration.
- **noquery** – prevents **ntpq** and **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](#) for more details), do not remove the **noquery** option from the **restrict default** command on publicly accessible systems.

22.16.2. Configure Rate Limiting Access to an NTP Service

To enable rate limiting access to the **NTP** service running on a system, add the **limited** option to the **restrict** command as explained in [Section 22.16.1, "Configure Access Control to an NTP Service"](#). If you do not want to use the default discard parameters, then also use the **discard** command as explained here.

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

```
discard [average value] [minimum value] [monitor value]
```

- **average** – specifies the minimum average packet spacing to be permitted, it accepts an argument in \log_2 seconds. The default value is 3 (2^3 equates to 8 seconds).
- **minimum** – specifies the minimum packet spacing to be permitted, it accepts an argument in \log_2 seconds. The default value is 1 (2^1 equates to 2 seconds).
- **monitor** – specifies the discard probability for packets once the permitted rate limits have been exceeded. The default value is 3000 seconds. This option is intended for servers that receive 1000 or more requests per second.

Examples of the **discard** command are as follows:

```
discard average 4
```

```
discard average 4 minimum 2
```

22.16.3. Adding a Peer Address

To add the address of a peer, that is to say, the address of a server running an **NTP** service of the same stratum, make use of the **peer** command in the **ntp.conf** file.

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

```
peer address
```

where *address* is an **IP** unicast address or a **DNS** resolvable name. The address must only be that of a system known to be a member of the same stratum. Peers should have at least one time source that is different to each other. Peers are normally systems under the same administrative control.

22.16.4. Adding a Server Address

To add the address of a server, that is to say, the address of a server running an **NTP** service of a higher stratum, make use of the **server** command in the **ntp.conf** file.

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

```
server address
```

where *address* is an **IP** unicast address or a **DNS** resolvable name. The address of a remote reference server or local reference clock from which packets are to be received.

22.16.5. Adding a Broadcast or Multicast Server Address

To add a broadcast or multicast address for sending, that is to say, the address to broadcast or multicast **NTP** packets to, make use of the **broadcast** command in the **ntp.conf** file.

The broadcast and multicast modes require authentication by default. See [Section 22.6, “Authentication Options for NTP”](#).

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

broadcast *address*

where *address* is an **IP** broadcast or multicast address to which packets are sent.

This command configures a system to act as an **NTP** broadcast server. The address used must be a broadcast or a multicast address. Broadcast address implies the **IPv4** address **255.255.255.255**. By default, routers do not pass broadcast messages. The multicast address can be an **IPv4** Class D address, or an **IPv6** address. The IANA has assigned **IPv4** multicast address **224.0.1.1** and **IPv6** address **FF05::101** (site local) to **NTP**. Administratively scoped **IPv4** multicast addresses can also be used, as described in [RFC 2365 Administratively Scoped IP Multicast](#).

22.16.6. Adding a Manycast Client Address

To add a manycast client address, that is to say, to configure a multicast address to be used for **NTP** server discovery, make use of the **manycastclient** command in the **ntp.conf** file.

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

manycastclient *address*

where *address* is an **IP** multicast address from which packets are to be received. The client will send a request to the address and select the best servers from the responses and ignore other servers. **NTP** communication then uses unicast associations, as if the discovered **NTP** servers were listed in **ntp.conf**.

This command configures a system to act as an **NTP** client. Systems can be both client and server at the same time.

22.16.7. Adding a Broadcast Client Address

To add a broadcast client address, that is to say, to configure a broadcast address to be monitored for broadcast **NTP** packets, make use of the **broadcastclient** command in the **ntp.conf** file.

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

broadcastclient

Enables the receiving of broadcast messages. Requires authentication by default. See [Section 22.6, “Authentication Options for NTP”](#).

This command configures a system to act as an **NTP** client. Systems can be both client and server at the same time.

22.16.8. Adding a Manycast Server Address

To add a unicast 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.

22.16.9. Adding a Multicast Client Address

To add a multicast client address, that is to say, to configure a multicast address to be monitored for multicast **NTP** packets, make use of the **multicastclient** command in the **ntp.conf** file.

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

multicastclient *address*

Enables the receiving of multicast messages. Where *address* is the address to subscribe to. This should be used together with authentication to prevent service disruption.

This command configures a system to act as an **NTP** client. Systems can be both client and server at the same time.

22.16.10. Configuring the Burst Option

Using the **burst** option against a public server is considered abuse. Do not use this option with public **NTP** servers. Use it only for applications within your own organization.

To increase the average quality of time offset statistics, add the following option to the end of a server command:

burst

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

22.16.11. Configuring the iburst Option

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

iburst

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

22.16.12. Configuring Symmetric Authentication Using a Key

To configure symmetric authentication using a key, add the following option to the end of a server or peer command:

key number

where *number* is in the range **1** to **65534** inclusive. This option enables the use of a *message authentication code* (MAC) in packets. This option is for use with the **peer**, **server**, **broadcast**, and **multicastclient** commands.

The option can be used in the **/etc/ntp.conf** file as follows:

```
server 192.168.1.1 key 10
broadcast 192.168.1.255 key 20
multicastclient 239.255.254.254 key 30
```

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

22.16.13. Configuring the Poll Interval

To change the default poll interval, add the following options to the end of a server or peer command:

minpoll value and **maxpoll value**

Options to change the default poll interval, where the interval in seconds will be calculated by raising 2 to the power of *value*, in other words, the interval is expressed in \log_2 seconds. The default **minpoll** value is 6, 2^6 equates to 64s. The default value for **maxpoll** is 10, which equates to 1024s. Allowed values are in the range 3 to 17 inclusive, which equates to 8s to 36.4h respectively. These options are for use with the **peer** or **server**. Setting a shorter **maxpoll** may improve clock accuracy.

22.16.14. Configuring Server Preference

To specify that a particular server should be preferred above others of similar statistical quality, add the following option to the end of a server or peer command:

prefer

Use this server for synchronization in preference to other servers of similar statistical quality. This option is for use with the **peer** or **server** commands.

22.16.15. Configuring the Time-to-Live for NTP Packets

To specify that a particular *time-to-live* (TTL) value should be used in place of the default, add the following option to the end of a server or peer command:

ttl value

Specify the time-to-live value to be used in packets sent by broadcast servers and multicast **NTP** servers. Specify the maximum time-to-live value to use for the “expanding ring search” by a multicast client. The default value is **127**.

22.16.16. Configuring the NTP Version to Use

To specify that a particular version of **NTP** should be used in place of the default, add the following option to the end of a server or peer command:

```
version value
```

Specify the version of **NTP** set in created **NTP** packets. The value can be in the range **1** to **4**. The default is **4**.

22.17. CONFIGURING THE HARDWARE CLOCK UPDATE

To configure the system clock to update the hardware clock, also known as the real-time clock (RTC), once after executing **ntpdate**, add the following line to **/etc/sysconfig/ntpdate**:

```
SYNC_HWCLOCK=yes
```

To update the hardware clock from the system clock, issue the following command as **root**:

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

When the system clock is being synchronized by **ntpd**, the kernel will in turn update the RTC every 11 minutes automatically.

22.18. CONFIGURING CLOCK SOURCES

To list the available clock sources on your system, issue the following commands:

```
~]$ cd /sys/devices/system/clocksource/clocksource0/  
clocksource0]$ cat available_clocksource  
kvm-clock tsc hpet acpi_pm  
clocksource0]$ cat current_clocksource  
kvm-clock
```

In the above example, the kernel is using **kvm-clock**. This was selected at boot time as this is a virtual machine.

To override the default clock source, add a line similar to the following in **grub.conf**:

```
clocksource=tsc
```

The available clock source is architecture dependent.

22.19. ADDITIONAL RESOURCES

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

22.19.1. Installed Documentation

- **ntpd(8)** man page – Describes **ntpd** in detail, including the command-line options.

- **ntp.conf(5)** man page – Contains information on how to configure associations with servers and peers.
- **ntpq(8)** man page – Describes the **NTP** query utility for monitoring and querying an **NTP** server.
- **ntpd(8)** man page – Describes the **ntpd** utility for querying and changing the state of **ntpd**.
- **ntp_auth(5)** man page – Describes authentication options, commands, and key management for **ntpd**.
- **ntp_keygen(8)** man page – Describes generating public and private keys for **ntpd**.
- **ntp_acc(5)** man page – Describes access control options using the **restrict** command.
- **ntp_mon(5)** man page – Describes monitoring options for the gathering of statistics.
- **ntp_clock(5)** man page – Describes commands for configuring reference clocks.
- **ntp_misc(5)** man page – Describes miscellaneous options.

22.19.2. Useful Websites

<http://doc.ntp.org/>

The NTP Documentation Archive

<http://www.eecis.udel.edu/~mills/ntp.html>

Network Time Synchronization Research Project.

<http://www.eecis.udel.edu/~mills/ntp/html/manopt.html>

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

CHAPTER 23. CONFIGURING PTP USING PTP4L

23.1. INTRODUCTION TO PTP

The *Precision Time Protocol* (PTP) is a protocol used to synchronize clocks in a network. When used in conjunction with hardware support, **PTP** is capable of sub-microsecond accuracy, which is far better than is normally obtainable with **NTP**. **PTP** support is divided between the kernel and user space. The kernel in Red Hat Enterprise Linux 6 now includes support for **PTP** clocks, which are provided by network drivers. The actual implementation of the protocol is known as **linuxptp**, a **PTPv2** implementation according to the IEEE standard 1588 for Linux.

The **linuxptp** package includes the **ptp4l** and **phc2sys** programs for clock synchronization. The **ptp4l** program implements the **PTP** boundary clock and ordinary clock. With hardware time stamping, it is used to synchronize the **PTP** hardware clock to the master clock, and with software time stamping it synchronizes the system clock to the master clock. The **phc2sys** program is needed only with hardware time stamping, for synchronizing the system clock to the **PTP** hardware clock on the *network interface card* (NIC).

23.1.1. Understanding PTP

The clocks synchronized by **PTP** are organized in a master-slave hierarchy. The slaves are synchronized to their masters which may be slaves to their own masters. The hierarchy is created and updated automatically by the *best master clock* (BMC) algorithm, which runs on every clock. When a clock has only one port, it can be *master* or *slave*, such a clock is called an *ordinary clock* (OC). A clock with multiple ports can be master on one port and slave on another, such a clock is called a *boundary clock* (BC). The top-level master is called the *grandmaster clock*, which can be synchronized by using a *Global Positioning System* (GPS) time source. By using a GPS-based time source, disparate networks can be synchronized with a high-degree of accuracy.

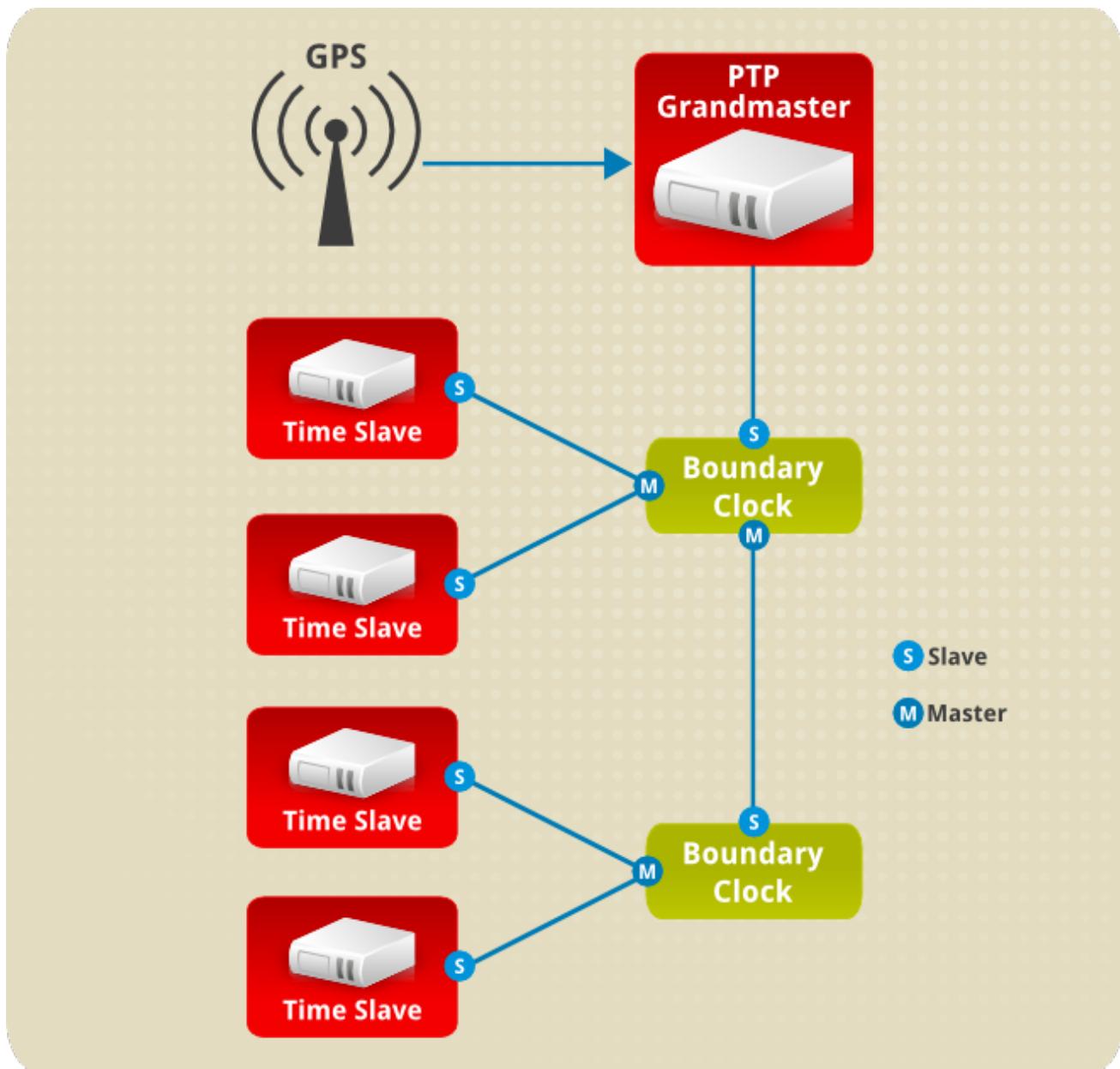


Figure 23.1. PTP grandmaster, boundary, and slave Clocks

23.1.2. Advantages of PTP

One of the main advantages that **PTP** has over the *Network Time Protocol* (NTP) is hardware support present in various *network interface controllers* (NIC) and network switches. This specialized hardware allows **PTP** to account for delays in message transfer, and greatly improves the accuracy of time synchronization. While it is possible to use non-PTP enabled hardware components within the network, this will often cause an increase in jitter or introduce an asymmetry in the delay resulting in synchronization inaccuracies, which add up with multiple non-PTP aware components used in the communication path. To achieve the best possible accuracy, it is recommended that all networking components between **PTP** clocks are **PTP** hardware enabled. Time synchronization in larger networks where not all of the networking hardware supports **PTP** might be better suited for **NTP**.

With hardware **PTP** support, the NIC has its own on-board clock, which is used to time stamp the received and transmitted **PTP** messages. It is this on-board clock that is synchronized to the **PTP** master, and the computer's system clock is synchronized to the **PTP** hardware clock on the NIC. With software **PTP** support, the system clock is used to time stamp the **PTP** messages and it is synchronized

to the **PTP** master directly. Hardware **PTP** support provides better accuracy since the NIC can time stamp the **PTP** packets at the exact moment they are sent and received while software **PTP** support requires additional processing of the **PTP** packets by the operating system.

23.2. USING PTP

In order to use **PTP**, the kernel network driver for the intended interface has to support either software or hardware time stamping capabilities.

23.2.1. Checking for Driver and Hardware Support

In addition to hardware time stamping support being present in the driver, the NIC must also be capable of supporting this functionality in the physical hardware. The best way to verify the time stamping capabilities of a particular driver and NIC is to use the **ethtool** utility to query the interface as follows:

```
~]# ethtool -T eth3
Time stamping parameters for eth3:
Capabilities:
    hardware-transmit  (SOF_TIMESTAMPING_TX_HARDWARE)
    software-transmit  (SOF_TIMESTAMPING_TX_SOFTWARE)
    hardware-receive   (SOF_TIMESTAMPING_RX_HARDWARE)
    software-receive   (SOF_TIMESTAMPING_RX_SOFTWARE)
    software-system-clock (SOF_TIMESTAMPING_SOFTWARE)
    hardware-raw-clock (SOF_TIMESTAMPING_RAW_HARDWARE)
PTP Hardware Clock: 0
Hardware Transmit Timestamp Modes:
    off          (HWTSTAMP_TX_OFF)
    on           (HWTSTAMP_TX_ON)
Hardware Receive Filter Modes:
    none         (HWTSTAMP_FILTER_NONE)
    all          (HWTSTAMP_FILTER_ALL)
```

Where *eth3* is the interface you want to check.

For software time stamping support, the parameters list should include:

- **SOF_TIMESTAMPING_SOFTWARE**
- **SOF_TIMESTAMPING_TX_SOFTWARE**
- **SOF_TIMESTAMPING_RX_SOFTWARE**

For hardware time stamping support, the parameters list should include:

- **SOF_TIMESTAMPING_RAW_HARDWARE**
- **SOF_TIMESTAMPING_TX_HARDWARE**
- **SOF_TIMESTAMPING_RX_HARDWARE**

23.2.2. Installing PTP

The kernel in Red Hat Enterprise Linux 6 now includes support for **PTP**. User space support is provided by the tools in the **linuxptp** package. To install **linuxptp**, issue the following command as **root**:

```
~]# yum install linuxptp
```

This will install **ptp4l** and **phc2sys**.

Do not run more than one service to set the system clock's time at the same time. If you intend to serve **PTP** time using **NTP**, see [Section 23.7, "Serving PTP Time With NTP"](#).

23.2.3. Starting ptp4l

The **ptp4l** program tries to use hardware time stamping by default. To use **ptp4l** with hardware time stamping capable drivers and NICs, you must provide the network interface to use with the **-i** option. Enter the following command as **root**:

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

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

```
~]# ptp4l -i eth3 -m
selected eth3 as PTP clock
port 1: INITIALIZING to LISTENING on INITIALIZE
port 0: INITIALIZING to LISTENING on INITIALIZE
port 1: new foreign master 00a069.ffe.0b552d-1
selected best master clock 00a069.ffe.0b552d
port 1: LISTENING to UNCALIBRATED on RS_SLAVE
master offset -23947 s0 freq +0 path delay 11350
master offset -28867 s0 freq +0 path delay 11236
master offset -32801 s0 freq +0 path delay 10841
master offset -37203 s1 freq +0 path delay 10583
master offset -7275 s2 freq -30575 path delay 10583
port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
master offset -4552 s2 freq -30035 path delay 10385
```

The master offset value is the measured offset from the master in nanoseconds. The **s0**, **s1**, **s2** strings indicate the different clock servo states: **s0** is unlocked, **s1** is clock step and **s2** is locked. Once the servo is in the locked state (**s2**), the clock will not be stepped (only slowly adjusted) unless the **pi_offset_const** option is set to a positive value in the configuration file (described in the [ptp4l\(8\)](#) man page). The **freq** value is the frequency adjustment of the clock in parts per billion (ppb). The path delay value is the estimated delay of the synchronization messages sent from the master in nanoseconds. Port 0 is a Unix domain socket used for local **PTP** management. Port 1 is the **eth3** interface (based on the example above.) INITIALIZING, LISTENING, UNCALIBRATED and SLAVE are some of possible port states which change on the INITIALIZE, RS_SLAVE, MASTER_CLOCK_SELECTED events. In the last state change message, the port state changed from UNCALIBRATED to SLAVE indicating successful synchronization with a **PTP** master clock.

The **ptp4l** program can also be started as a service by running:

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

When running as a service, options are specified in the **/etc/sysconfig/ptp4l** file. More information on the different **ptp4l** options and the configuration file settings can be found in the [ptp4l\(8\)](#) man page.

By default, messages are sent to **/var/log/messages**. However, specifying the **-m** option enables logging to standard output which can be useful for debugging purposes.

To enable software time stamping, the **-S** option needs to be used as follows:

```
~]# ptpt4l -i eth3 -m -S
```

23.2.3.1. Selecting a Delay Measurement Mechanism

There are two different delay measurement mechanisms and they can be selected by means of an option added to the **ptpt4l** 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 **ptpt4l** in E2E mode. It will change to P2P mode if a peer delay request is received.



NOTE

All clocks on a single **PTP** communication path must use the same mechanism to measure the delay. A warning will be printed when a peer delay request is received on a port using the E2E mechanism. A warning will be printed when a E2E delay request is received on a port using the P2P mechanism.

23.3. SPECIFYING A CONFIGURATION FILE

The command-line options and other options, which cannot be set on the command line, can be set in an optional configuration file.

No configuration file is read by default, so it needs to be specified at runtime with the **-f** option. For example:

```
~]# ptpt4l -f /etc/ptpt4l.conf
```

A configuration file equivalent to the **-i eth3 -m -S** options shown above would look as follows:

```
~]# cat /etc/ptpt4l.conf
[global]
verbose      1
```

```
time_stamping      software
[eth3]
```

23.4. USING THE PTP MANAGEMENT CLIENT

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

```
~]# pmc -u -b 0 'GET CURRENT_DATA_SET'
sending: GET CURRENT_DATA_SET
90e2ba.ffe.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.ffe.20c7f8-0 seq 0 RESPONSE MANAGMENT TIME_STATUS_NP
  master_offset      310
  ingress_time       1361545089345029441
  cumulativeScaledRateOffset +1.000000000
  scaledLastGmPhaseChange 0
  gmTimeBaseIndicator 0
  lastGmPhaseChange   0x0000'0000000000000000.0000
  gmPresent          true
  gmIdentity         00a069.ffe.0b552d
```

Setting the **-b** option to **zero** limits the boundary to the locally running **ptp4l** instance. A larger boundary value will retrieve the information also from **PTP** nodes further from the local clock. The retrievable information includes:

- **stepsRemoved** is the number of communication paths to the grandmaster clock.
- **offsetFromMaster** and **master_offset** is the last measured offset of the clock from the master in nanoseconds.
- **meanPathDelay** is the estimated delay of the synchronization messages sent from the master in nanoseconds.
- if **gmPresent** is true, the **PTP** clock is synchronized to a master, the local clock is not the grandmaster clock.
- **gmIdentity** is the grandmaster's identity.

For a full list of **pmc** commands, type the following as **root**:

```
~]# pmc help
```

Additional information is available in the **pmc(8)** man page.

23.5. SYNCHRONIZING THE CLOCKS

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

OPTIONS="-a -r"

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

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

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

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

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

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

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

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

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

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

Normally, **PTP** operates in the *International Atomic Time* (TAI) timescale, while the system clock is kept in *Coordinated Universal Time* (UTC). The current offset between the TAI and UTC timescales is 36 seconds. The offset changes when leap seconds are inserted or deleted, which typically happens every few years. The **-O** option needs to be used to set this offset manually when the **-w** is not used, as follows:

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

Once the **phc2sys** servo is in a locked state, the clock will not be stepped, unless the **-S** option is used. This means that the **phc2sys** program should be started after the **ptp4l** program has synchronized the **PTP** hardware clock. However, with **-w**, it is not necessary to start **phc2sys** after **ptp4l** as it will wait for it to synchronize the clock.

The **phc2sys** program can also be started as a service by running:

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

When running as a service, options are specified in the **/etc/sysconfig/phc2sys** file. More information on the different **phc2sys** options can be found in the **phc2sys(8)** man page.

Note that the examples in this section assume the command is run on a slave system or slave port.

23.6. VERIFYING TIME SYNCHRONIZATION

When **PTP** time synchronization is working properly, new messages with offsets and frequency

adjustments will be printed periodically to the **ptp4l** and **phc2sys** (if hardware time stamping is used) outputs. These values will eventually converge after a short period of time. These messages can be seen in **/var/log/messages** file. An example of the **ptp4l** output follows:

```
ptp4l[352.359]: selected /dev/ptp0 as PTP clock
ptp4l[352.361]: port 1: INITIALIZING to LISTENING on INITIALIZE
ptp4l[352.361]: port 0: INITIALIZING to LISTENING on INITIALIZE
ptp4l[353.210]: port 1: new foreign master 00a069.ffe.0b552d-1
ptp4l[357.214]: selected best master clock 00a069.ffe.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.ffe.0b552d-1
ptp4l: [619.574] selected best master clock 00a069.ffe.0b552d
ptp4l: [619.574] port 1: LISTENING to UNCALIBRATED on RS_SLAVE
ptp4l: [623.573] port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l: [684.649] rms 669 max 3691 freq -29383 ± 3735 delay 9232 ± 122
ptp4l: [748.724] rms 253 max 588 freq -29787 ± 221 delay 9219 ± 158
ptp4l: [812.793] rms 287 max 673 freq -29802 ± 248 delay 9211 ± 183
ptp4l: [876.853] rms 226 max 534 freq -29795 ± 197 delay 9221 ± 138
ptp4l: [940.925] rms 250 max 562 freq -29801 ± 218 delay 9199 ± 148
ptp4l: [1004.988] rms 226 max 525 freq -29802 ± 196 delay 9228 ± 143
ptp4l: [1069.065] rms 300 max 646 freq -29802 ± 259 delay 9214 ± 176
ptp4l: [1133.125] rms 226 max 505 freq -29792 ± 197 delay 9225 ± 159
ptp4l: [1197.185] rms 244 max 688 freq -29790 ± 211 delay 9201 ± 162
```

To reduce the output from the **phc2sys**, it can be called it with the **-u** option as follows:

```
~]# phc2sys -u summary-updates
```

Where *summary-updates* is the number of clock updates to include in summary statistics. An example follows:

```
~]# phc2sys -s eth3 -w -m -u 60
phc2sys[700.948]: rms 1837 max 10123 freq -36474 ± 4752 delay 2752 ± 16
phc2sys[760.954]: rms 194 max 457 freq -37084 ± 174 delay 2753 ± 12
phc2sys[820.963]: rms 211 max 487 freq -37085 ± 185 delay 2750 ± 19
phc2sys[880.968]: rms 183 max 440 freq -37102 ± 164 delay 2734 ± 91
phc2sys[940.973]: rms 244 max 584 freq -37095 ± 216 delay 2748 ± 16
phc2sys[1000.979]: rms 220 max 573 freq -36666 ± 182 delay 2747 ± 43
phc2sys[1060.984]: rms 266 max 675 freq -36759 ± 234 delay 2753 ± 17
```

23.7. SERVING PTP TIME WITH NTP

The **ntpd** daemon can be configured to distribute the time from the system clock synchronized by **ptp4l** or **phc2sys** by using the LOCAL reference clock driver. To prevent **ntpd** from adjusting the system clock, the **ntp.conf** file must not specify any **NTP** servers. The following is a minimal example of **ntp.conf**:

```
~]# cat /etc/ntp.conf
server 127.127.1.0
fudge 127.127.1.0 stratum 0
```



NOTE

When the **DHCP** client program, **dhclient**, receives a list of **NTP** servers from the **DHCP** server, it adds them to **ntp.conf** and restarts the service. To disable that feature, add **PEERNTP=no** to **/etc/sysconfig/network**.

23.8. SERVING NTP TIME WITH PTP

NTP to **PTP** synchronization in the opposite direction is also possible. When **ntpd** is used to synchronize the system clock, **ptp4l** can be configured with the **priority1** option (or other clock options included in the best master clock algorithm) to be the grandmaster clock and distribute the time from the system clock via **PTP**:

```
~]# cat /etc/ptp4l.conf
[global]
priority1 127
[eth3]
# ptp4l -f /etc/ptp4l.conf
```

With hardware time stamping, **phc2sys** needs to be used to synchronize the **PTP** hardware clock to the system clock:

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

To prevent quick changes in the **PTP** clock's frequency, the synchronization to the system clock can be loosened by using smaller **P** (proportional) and **I** (integral) constants of the PI servo:

```
~]# phc2sys -c eth3 -s CLOCK_REALTIME -w -P 0.01 -I 0.0001
```

23.9. SYNCHRONIZE TO PTP OR NTP TIME USING TIMEMASTER

When there are multiple **PTP** domains available on the network, or fallback to **NTP** is needed, the **timemaster** program can be used to synchronize the system clock to all available time sources. The **PTP** time is provided by **phc2sys** and **ptp4l** via *shared memory driver* (SHM reference clocks to **chrony** 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 **chrony** or **ntpd**, and starts the **ptp4l**, **phc2sys**, and **chrony** or **ntpd** processes as needed. It will remove the generated configuration files on exit. It writes configuration files for **chrony**, **ntpd**, and **ptp4l** to **/var/run/timemaster/**.

23.9.1. Starting timemaster as a Service

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

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

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

23.9.2. Understanding the timemaster Configuration File

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

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

```
~]$ less /etc/timemaster.conf
```

```
# Configuration file for timemaster

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

#[ptp_domain 0]
#interfaces eth0

[timemaster]
ntp_program chronyd

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

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

[ptp4l.conf]

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

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

[phc2sys]
path /usr/sbin/phc2sys

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

Notice the section named as follows:

[ntp_server address]

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

Notice the section named as follows:

[ptp_domain number]

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

An instance of **ptp4l** is started for every interface which has its own **PTP** clock and hardware time stamping is enabled automatically. Interfaces that support hardware time stamping have a **PTP** clock

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

Notice the section named as follows:

```
[timemaster]
```

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

The section headings are as follows:

- **[ntp_server ntp-server.local]** – Specify polling intervals for this server. Create additional sections as required. Include the host name or **IP** address in the section heading.
- **[ptp_domain 0]** – Specify interfaces that have **PTP** clocks configured for this domain. Create additional sections with, the appropriate domain number, as required.
- **[timemaster]** – Specify the **NTP** daemon to be used. Possible values are **chronyd** and **ntpd**.
- **[chrony.conf]** – Specify any additional settings to be copied to the configuration file generated for **chronyd**.
- **[ntp.conf]** – Specify any additional settings to be copied to the configuration file generated for **ntpd**.
- **[ptp4l.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.

23.9.3. Configuring timemaster Options

Procedure 23.1. Editing the timemaster Configuration File

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

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

2. For each **NTP** server you want to control using **timemaster**, create **[ntp_server address]** sections . Note that the short polling values in the example section are not suitable for a public server, see [Chapter 22, Configuring NTP Using ntpd](#) for an explanation of suitable **minpoll** and **maxpoll** values.

3. To add interfaces that should be used in a domain, edit the **#[ptp_domain 0]** section and add the interfaces. Create additional domains as required. For example:

```
[ptp_domain 0]
interfaces eth0
```

```
[ptp_domain 1]
interfaces eth1
```

4. If required to use **ntpd** as the **NTP** daemon on this system, change the default entry in the **[timemaster]** section from **chrony****d** to **ntpd**. See Configuring NTP Using the chrony Suite for information on the differences between **ntpd** and **chrony****d**.
5. If using **chrony****d** 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 **[ptp4l.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 **[chrony****d****]** section, add any command line options to be passed to **chrony****d** when called by **timemaster**. See Configuring NTP Using the chrony Suite for information on using **chrony****d**.
9. In the **[ntpd]** section, add any command line options to be passed to **ntpd** when called by **timemaster**. See [Chapter 22, Configuring NTP Using ntpd](#) for information on using **ntpd**.
10. In the **[phc2sys]** section, add any command line options to be passed to **phc2sys** when called by **timemaster**. This chapter documents common options and more information is available in the **phy2sys(8)** manual page.
11. In the **[ptp4l]** section, add any command line options to be passed to **ptp4l** when called by **timemaster**. This chapter documents common options and more information is available in the **ptp4l(8)** manual page.
12. Save the configuration file and restart **timemaster** by issuing the following command as **root**:

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

23.10. IMPROVING ACCURACY

Previously, test results indicated that disabling the tickless kernel capability could significantly improve the stability of the system clock, and thus improve the **PTP** synchronization accuracy (at the cost of increased power consumption). The kernel tickless mode can be disabled by adding **nohz=off** to the kernel boot option parameters. However, recent improvements applied to **kernel-3.10.0-197.el7** have greatly improved the stability of the system clock and the difference in stability of the clock with and without **nohz=off** should be much smaller now for most users.

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

```
clock_servo linreg
```

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

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

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

```
-E linreg
```

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

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

23.11. ADDITIONAL RESOURCES

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

23.11.1. Installed Documentation

- **ptp4l(8)** man page – Describes **ptp4l** options including the format of the configuration file.
- **pmc(8)** man page – Describes the **PTP** management client and its command options.
- **phc2sys(8)** man page – Describes a tool for synchronizing the system clock to a **PTP** hardware clock (PHC).

23.11.2. Useful Websites

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

The Linux PTP project.

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

The IEEE 1588 Standard.

PART VII. MONITORING AND AUTOMATION

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

CHAPTER 24. SYSTEM MONITORING TOOLS

In order to configure the system, system administrators often need to determine the amount of free memory, how much free disk space is available, how the hard drive is partitioned, or what processes are running.

24.1. VIEWING SYSTEM PROCESSES

24.1.1. Using the ps Command

The **ps** command allows you to display information about running processes. It produces a static list, that is, a snapshot of what is running when you execute the command. If you want a constantly updated list of running processes, use the **top** command or the **System Monitor** application instead.

To list all processes that are currently running on the system including processes owned by other users, type the following at a shell prompt:

```
ps ax
```

For each listed process, the **ps ax** command displays the process ID (**PID**), the terminal that is associated with it (**TTY**), the current status (**STAT**), the cumulated CPU time (**TIME**), and the name of the executable file (**COMMAND**). For example:

```
~]$ ps ax
PID TTY STAT TIME COMMAND
 1 ? Ss 0:01 /sbin/init
 2 ? S 0:00 [kthreadd]
 3 ? S 0:00 [migration/0]
 4 ? S 0:00 [ksoftirqd/0]
 5 ? S 0:00 [migration/0]
 6 ? S 0:00 [watchdog/0]
[output truncated]
```

To display the owner alongside each process, use the following command:

```
ps aux
```

Apart from the information provided by the **ps ax** command, **ps aux** displays the effective user name of the process owner (**USER**), the percentage of the CPU (**%CPU**) and memory (**%MEM**) usage, the virtual memory size in kilobytes (**VSZ**), the non-swapped physical memory size in kilobytes (**RSS**), and the time or date the process was started. For instance:

```
~]$ ps aux
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
root 1 0.0 0.1 19404 832 ? S Mar02 0:01 /sbin/init
root 2 0.0 0.0 0 0 ? S Mar02 0:00 [kthreadd]
root 3 0.0 0.0 0 0 ? S Mar02 0:00 [migration/0]
root 4 0.0 0.0 0 0 ? S Mar02 0:00 [ksoftirqd/0]
root 5 0.0 0.0 0 0 ? S Mar02 0:00 [migration/0]
root 6 0.0 0.0 0 0 ? R Mar02 0:00 [watchdog/0]
[output truncated]
```

You can also use the **ps** command in a combination with **grep** to see if a particular process is running. For example, to determine if **Emacs** is running, type:

```
~]$ ps ax | grep emacs
12056 pts/3 S+ 0:00 emacs
12060 pts/2 S+ 0:00 grep --color=auto emacs
```

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

24.1.2. Using the top Command

The **top** command displays a real-time list of processes that are running on the system. It also displays additional information about the system uptime, current CPU and memory usage, or total number of running processes, and allows you to perform actions such as sorting the list or killing a process.

To run the **top** command, type the following at a shell prompt:

```
top
```

For each listed process, the **top** command displays the process ID (**PID**), the effective user name of the process owner (**USER**), the priority (**PR**), the nice value (**NI**), the amount of virtual memory the process uses (**VIRT**), the amount of non-swapped physical memory the process uses (**RES**), the amount of shared memory the process uses (**SHR**), the process status field **S**, the percentage of the CPU (**%CPU**) and memory (**%MEM**) usage, the accumulated CPU time (**TIME+**), and the name of the executable file (**COMMAND**). For example:

```
~]$ top
top - 02:19:11 up 4 days, 10:37, 5 users, load average: 0.07, 0.13, 0.09
Tasks: 160 total, 1 running, 159 sleeping, 0 stopped, 0 zombie
Cpu(s): 10.7%us, 1.0%sy, 0.0%ni, 88.3%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 760752k total, 644360k used, 116392k free, 3988k buffers
Swap: 1540088k total, 76648k used, 1463440k free, 196832k cached

 PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
14401 jhradile 20 0 313m 10m 5732 S 5.6 1.4 6:27.29 gnome-system-mo
1764 root 20 0 133m 23m 4756 S 5.3 3.2 6:32.66 Xorg
13865 jhradile 20 0 1625m 177m 6628 S 0.7 23.8 0:57.26 java
  20 root 20 0 0 0 S 0.3 0.0 4:44.39 ata/0
2085 root 20 0 40396 348 276 S 0.3 0.0 1:57.13 udisks-daemon
  1 root 20 0 19404 832 604 S 0.0 0.1 0:01.21 init
  2 root 20 0 0 0 S 0.0 0.0 0:00.01 kthreadd
  3 root RT 0 0 0 S 0.0 0.0 0:00.00 migration/0
  4 root 20 0 0 0 S 0.0 0.0 0:00.02 ksoftirqd/0
  5 root RT 0 0 0 S 0.0 0.0 0:00.00 migration/0
  6 root RT 0 0 0 S 0.0 0.0 0:00.00 watchdog/0
  7 root 20 0 0 0 S 0.0 0.0 0:01.00 events/0
  8 root 20 0 0 0 S 0.0 0.0 0:00.00 cpuset
  9 root 20 0 0 0 S 0.0 0.0 0:00.00 khelper
 10 root 20 0 0 0 S 0.0 0.0 0:00.00 netns
 11 root 20 0 0 0 S 0.0 0.0 0:00.00 async/mgr
 12 root 20 0 0 0 S 0.0 0.0 0:00.00 pm
[output truncated]
```

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

Table 24.1. Interactive top commands

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

24.1.3. Using the System Monitor Tool

The **Processes** tab of the **System Monitor** tool allows you to view, search for, change the priority of, and kill processes from the graphical user interface. To install the tool, issue the following command as **root**:

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

To start the **System Monitor** tool, either select **Applications → System Tools → System Monitor** from the panel, or type **gnome-system-monitor** at a shell prompt. Then click the **Processes** tab to view the list of running processes.

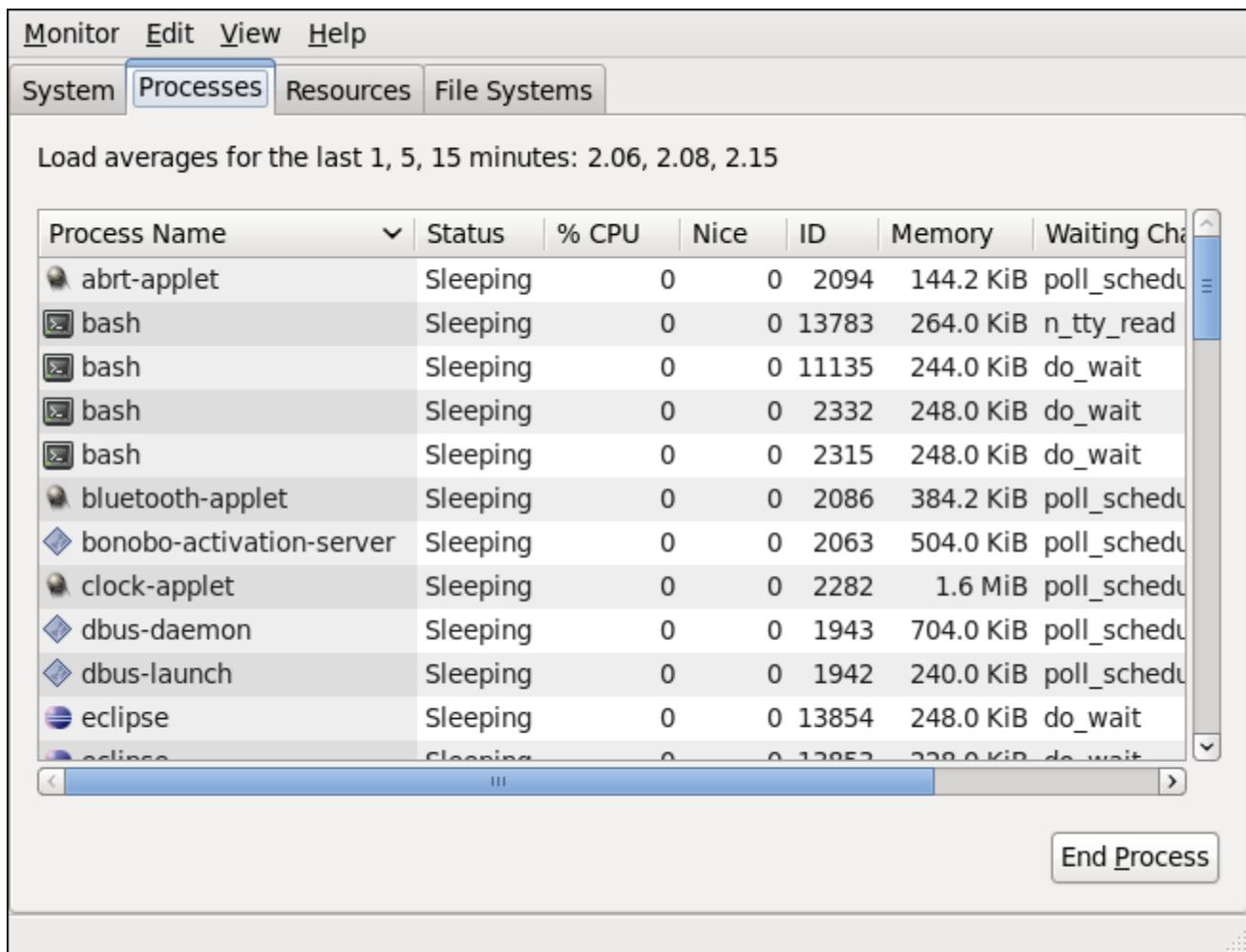


Figure 24.1. System Monitor – Processes

For each listed process, the **System Monitor** tool displays its name (**Process Name**), current status (**Status**), percentage of the CPU usage (**% CPU**), nice value (**Nice**), process ID (**ID**), memory usage (**Memory**), the channel the process is waiting in (**Waiting Channel**), and additional details about the session (**Session**). To sort the information by a specific column in ascending order, click the name of that column. Click the name of the column again to toggle the sort between ascending and descending order.

By default, the **System Monitor** tool displays a list of processes that are owned by the current user. Selecting various options from the **View** menu allows you to:

- view only active processes,
- view all processes,
- view your processes,
- view process dependencies,
- view a memory map of a selected process,
- view the files opened by a selected process, and
- refresh the list of processes.

Additionally, various options in the **Edit** menu allows you to:

- stop a process,
- continue running a stopped process,
- end a process,
- kill a process,
- change the priority of a selected process, and
- edit the **System Monitor** preferences, such as the refresh interval for the list of processes, or what information to show.

You can also end a process by selecting it from the list and clicking the **End Process** button.

24.2. VIEWING MEMORY USAGE

24.2.1. Using the free Command

The **free** command allows you to display the amount of free and used memory on the system. To do so, type the following at a shell prompt:

```
free
```

The **free** command provides information about both the physical memory (**Mem**) and swap space (**Swap**). It displays the total amount of memory (**total**), as well as the amount of memory that is in use (**used**), free (**free**), shared (**shared**), in kernel buffers (**buffers**), and cached (**cached**). For example:

```
~]$ free
      total    used     free   shared   buffers   cached
Mem:    760752   661332   99420       0    6476   317200
      -/+ buffers/cache:  337656   423096
Swap:   1540088   283652  1256436
```

By default, **free** displays the values in kilobytes. To display the values in megabytes, supply the **-m** command-line option:

```
free -m
```

For instance:

```
~]$ free -m
      total    used     free   shared   buffers   cached
Mem:     742     646      96       0       6     309
      -/+ buffers/cache:   330     412
Swap:   1503     276    1227
```

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

24.2.2. Using the System Monitor Tool

The **Resources** tab of the **System Monitor** tool allows you to view the amount of free and used memory on the system.

To start the **System Monitor** tool, either select **Applications → System Tools → System Monitor** from the panel, or type **gnome-system-monitor** at a shell prompt. Then click the **Resources** tab to view the system's memory usage.

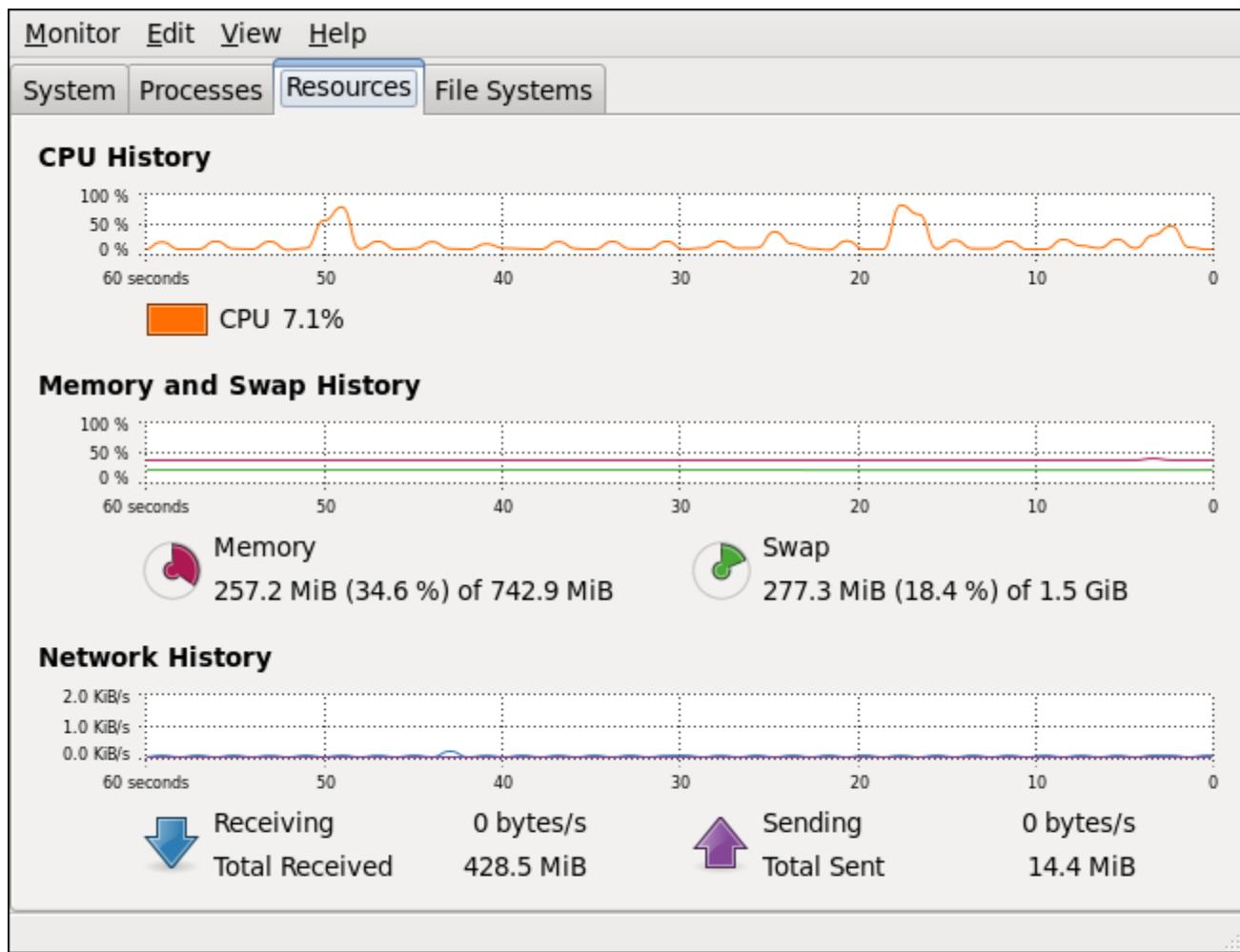


Figure 24.2. System Monitor – Resources

In the **Memory and Swap History** section, the **System Monitor** tool displays a graphical representation of the memory and swap usage history, as well as the total amount of the physical memory (**Memory**) and swap space (**Swap**) and how much of it is in use.

24.3. VIEWING CPU USAGE

24.3.1. Using the System Monitor Tool

The **Resources** tab of the **System Monitor** tool allows you to view the current CPU usage on the system.

To start the **System Monitor** tool, either select **Applications → System Tools → System Monitor** from the panel, or type **gnome-system-monitor** at a shell prompt. Then click the **Resources** tab to view the system's CPU usage.

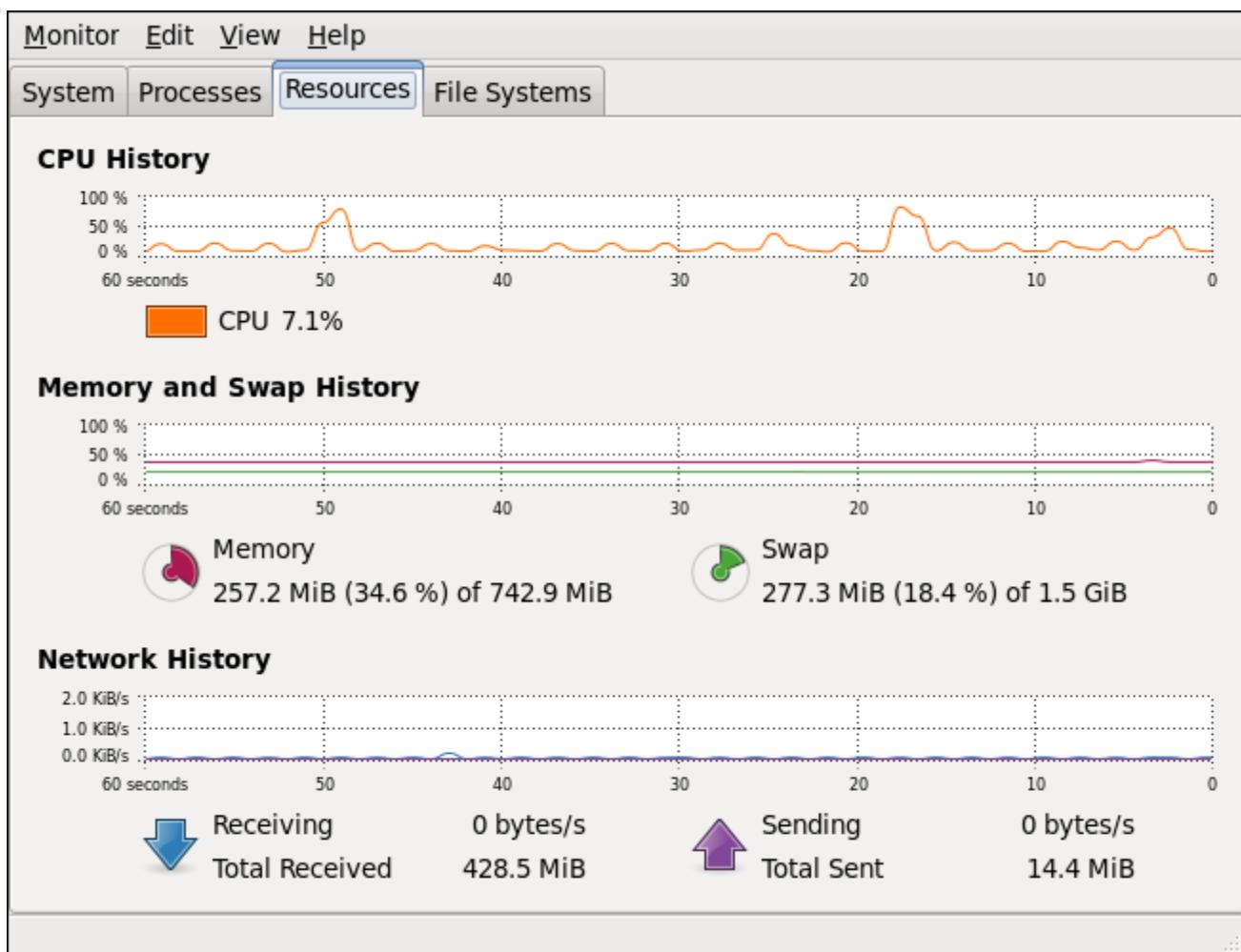


Figure 24.3. System Monitor – Resources

In the **CPU History** section, the **System Monitor** tool displays a graphical representation of the CPU usage history and shows the percentage of how much CPU is currently in use.

24.4. VIEWING BLOCK DEVICES AND FILE SYSTEMS

24.4.1. Using the lsblk Command

The **lsblk** command allows you to display a list of available block devices. To do so, type the following at a shell prompt:

```
lsblk
```

For each listed block device, the **lsblk** command displays the device name (**NAME**), major and minor device number (**MAJ:MIN**), if the device is removable (**RM**), what is its size (**SIZE**), if the device is read-only (**RO**), what type is it (**TYPE**), and where the device is mounted (**MOUNTPOINT**). For example:

```
~]$ lsblk
NAME      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0       11:0    1 1024M 0 rom
vda      252:0    0  20G 0 rom
└-vda1    252:1    0 500M 0 part /boot
└-vda2    252:2    0 19.5G 0 part
  |-vg_kvm-lv_root (dm-0) 253:0  0 18G 0 lvm /
  `|-vg_kvm-lv_swap (dm-1) 253:1  0 1.5G 0 lvm [SWAP]
```

By default, **lsblk** lists block devices in a tree-like format. To display the information as an ordinary list, add the **-l** command-line option:

lsblk -l

For instance:

```
~]$ lsblk -l
NAME      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0        11:0    1 1024M 0 rom
vda       252:0    0  20G 0 rom
vda1      252:1    0 500M 0 part /boot
vda2      252:2    0 19.5G 0 part
vg_kvm-lv_root (dm-0) 253:0  0 18G 0 lvm /
vg_kvm-lv_swap (dm-1) 253:1  0 1.5G 0 lvm [SWAP]
```

For a complete list of available command-line options, see the **lsblk(8)** manual page.

24.4.2. Using the blkid Command

The **blkid** command allows you to display information about available block devices. To do so, type the following at a shell prompt as **root**:

blkid

For each listed block device, the **blkid** command displays available attributes such as its *universally unique identifier (UUID)*, file system type (**TYPE**), or volume label (**LABEL**). For example:

```
~]$ blkid
/dev/vda1: UUID="7fa9c421-0054-4555-b0ca-b470a97a3d84" TYPE="ext4"
/dev/vda2: UUID="7lvYzk-TnnK-oPjf-ipdD-cofz-DXAJ-gPdgBW" TYPE="LVM2_member"
/dev/mapper/vg_kvm-lv_root: UUID="a07b967c-71a0-4925-ab02-aebcad2ae824" TYPE="ext4"
/dev/mapper/vg_kvm-lv_swap: UUID="d7ef54ca-9c41-4de4-ac1b-4193b0c1ddb6" TYPE="swap"
```

By default, the **blkid** command lists all available block devices. To display information about a particular device only, specify the device name on the command line:

blkid *device_name*

For instance, to display information about **/dev/vda1**, type:

```
~]$ blkid /dev/vda1
/dev/vda1: UUID="7fa9c421-0054-4555-b0ca-b470a97a3d84" TYPE="ext4"
```

You can also use the above command with the **-p** and **-o udev** command-line options to obtain more detailed information. Note that **root** privileges are required to run this command:

blkid -po udev *device_name*

For example:

```
~]$ blkid -po udev /dev/vda1
```

```
ID_FS_UUID=7fa9c421-0054-4555-b0ca-b470a97a3d84
ID_FS_UUID_ENC=7fa9c421-0054-4555-b0ca-b470a97a3d84
ID_FS_VERSION=1.0
ID_FS_TYPE=ext4
ID_FS_USAGE=filesystem
```

For a complete list of available command-line options, see the **blkid(8)** manual page.

24.4.3. Using the **findmnt** Command

The **findmnt** command allows you to display a list of currently mounted file systems. To do so, type the following at a shell prompt:

findmnt

For each listed file system, the **findmnt** command displays the target mount point (**TARGET**), source device (**SOURCE**), file system type (**FSTYPE**), and relevant mount options (**OPTIONS**). For example:

```
~]$ findmnt
TARGET           SOURCE          FSTYPE   OPTIONS
/               /dev/mapper/vg_kvm-lv_root ext4    rw,relatime,sec
|/proc           /proc            proc     rw,relatime
||-/proc/bus/usb /proc/bus/usb   usbfs   rw,relatime
||`-/proc/sys/fs/binfmt_misc      binfmt_m   rw,relatime
|-/sys           /sys             sysfs   rw,relatime,sec
|-selinux        selinux         selinuxf rw,relatime
|-/dev            udev             devtmpfs rw,relatime,sec
|`-/dev          udev             devtmpfs rw,relatime,sec
|  |-/dev/pts     devpts          devpts   rw,relatime,sec
|  `-/dev/shm     tmpfs            tmpfs   rw,relatime,sec
|-/boot          /dev/vda1       ext4    rw,relatime,sec
|-/var/lib/nfs/rpc_pipefs        sunrpc          rpc_pipe   rw,relatime
|-/misc          /etc/auto.misc  autoofs  rw,relatime,fd=
`-/net           -hosts          autoofs  rw,relatime,fd=
[output truncated]
```

By default, **findmnt** lists file systems in a tree-like format. To display the information as an ordinary list, add the **-l** command-line option:

findmnt -l

For instance:

```
~]$ findmnt -l
TARGET           SOURCE          FSTYPE   OPTIONS
/proc           /proc            proc     rw,relatime
/sys            /sys             sysfs   rw,relatime,seclabe
/dev            udev             devtmpfs rw,relatime,seclabe
/dev/pts         devpts          devpts   rw,relatime,seclabe
/dev/shm         tmpfs            tmpfs   rw,relatime,seclabe
/               /dev/mapper/vg_kvm-lv_root ext4    rw,relatime,seclabe
/selinux        selinux         selinuxf rw,relatime
/dev            udev             devtmpfs rw,relatime,seclabe
/proc/bus/usb   /proc/bus/usb   usbfs   rw,relatime
```

```

/boot          /dev/vda1      ext4  rw,relatime,seclabe
/proc/sys/fs/binfmt_misc binfmt_m rw,relatime
/var/lib/nfs/rpc_pipefs sunrpc  rpc_pipe rw,relatime
/misc          /etc/auto.misc autofs  rw,relatime,fd=7,pg
/net           -hosts        autofs  rw,relatime,fd=13,p
[output truncated]

```

You can also choose to list only file systems of a particular type. To do so, add the **-t** command-line option followed by a file system type:

findmnt -t type

For example, to list all **ext4** file systems, type:

```

~]$ findmnt -t ext4
TARGET SOURCE          FSTYPE OPTIONS
/   /dev/mapper/vg_kvm-lv_root ext4  rw,relatime,seclabel,barrier=1,data=ord
/boot /dev/vda1         ext4  rw,relatime,seclabel,barrier=1,data=ord

```

For a complete list of available command-line options, see the **findmnt(8)** manual page.

24.4.4. Using the df Command

The **df** command allows you to display a detailed report on the system's disk space usage. To do so, type the following at a shell prompt:

df

For each listed file system, the **df** command displays its name (**Filesystem**), size (**1K-blocks** or **Size**), how much space is used (**Used**), how much space is still available (**Available**), the percentage of space usage (**Use%**), and where is the file system mounted (**Mounted on**). For example:

```

~]$ df
Filesystem      1K-blocks  Used Available Use% Mounted on
/dev/mapper/vg_kvm-lv_root 18618236  4357360  13315112 25% /
tmpfs          380376    288  380088  1% /dev/shm
/dev/vda1       495844   77029  393215  17% /boot

```

By default, the **df** command shows the partition size in 1 kilobyte blocks and the amount of used and available disk space in kilobytes. To view the information in megabytes and gigabytes, supply the **-h** command-line option, which causes **df** to display the values in a human-readable format:

df -h

For instance:

```

~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/vg_kvm-lv_root 18G  4.2G  13G 25% /
tmpfs          372M 288K 372M  1% /dev/shm
/dev/vda1       485M 76M 384M 17% /boot

```

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

24.4.5. Using the du Command

The **du** command allows you to displays the amount of space that is being used by files in a directory. To display the disk usage for each of the subdirectories in the current working directory, run the command with no additional command-line options:

du

For example:

```
~]$ du  
14972 ./Downloads  
4     ./gnome2  
4     ./mozilla/extensions  
4     ./mozilla/plugins  
12     ./mozilla  
15004 .
```

By default, the **du** command displays the disk usage in kilobytes. To view the information in megabytes and gigabytes, supply the **-h** command-line option, which causes the utility to display the values in a human-readable format:

du -h

For instance:

```
~]$ du -h  
15M   ./Downloads  
4.0K   ./gnome2  
4.0K   ./mozilla/extensions  
4.0K   ./mozilla/plugins  
12K   ./mozilla  
15M   .
```

At the end of the list, the **du** command always shows the grand total for the current directory. To display only this information, supply the **-s** command-line option:

du -sh

For example:

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

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

24.4.6. Using the System Monitor Tool

The **File Systems** tab of the **System Monitor** tool allows you to view file systems and disk space usage in the graphical user interface.

To start the **System Monitor** tool, either select **Applications → System Tools → System Monitor** from the panel, or type **gnome-system-monitor** at a shell prompt. Then click the **File Systems** tab to view a list of file systems.

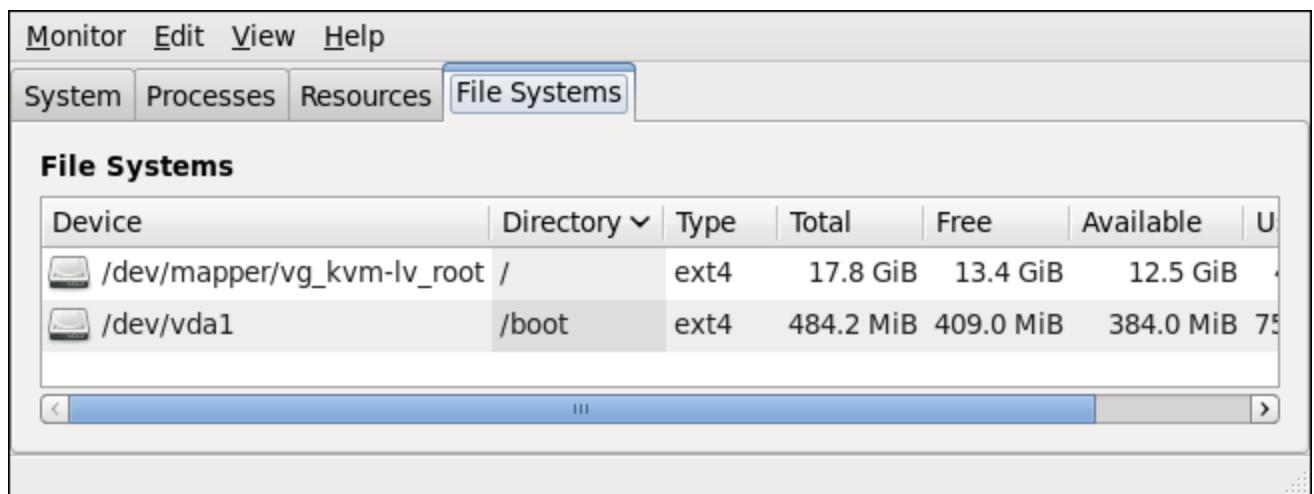


Figure 24.4. System Monitor – File Systems

For each listed file system, the **System Monitor** tool displays the source device (**Device**), target mount point (**Directory**), and file system type (**Type**), as well as its size (**Total**) and how much space is free (**Free**), available (**Available**), and used (**Used**).

24.4.7. Monitoring Files and Directories with gamin

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

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

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

- **/etc/gamin/gaminrc**
- **\$HOME/.gaminrc**
- **/etc/gamin/mandatory_gaminrc**

The configuration file accepts only the following commands:

Commands accepted by the configuration file

notify

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

poll

To express that polling should be used for matching paths.

fset

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

An example of such configuration file can be seen here:

```
# configuration for gamin
# Can be used to override the default behaviour.
# notify filepath(s) : indicate to use kernel notification
# poll filepath(s) : indicate to use polling instead
# fset fsname method poll_limit : indicate what method of notification for the file system
#           kernel - use the kernel for notification
#           poll - use polling for notification
#           none - don't use any notification
#           the poll_limit is the number of seconds
#           that must pass before a resource is polled again.
#           It is optional, and if it is not present the previous
#           value will be used or the default.
notify /mnt/local* /mnt/pictures* # use kernel notification on these paths
poll /temp/*                  # use poll notification on these paths
fset nfs poll 10               # use polling on nfs mounts and poll once every 10 seconds
```

The three configuration files are loaded in this order:

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

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

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

24.5. VIEWING HARDWARE INFORMATION

24.5.1. Using the lspci Command

The **lspci** command allows you to display information about PCI buses and devices that are attached to them. To list all PCI devices that are in the system, type the following at a shell prompt:

`lspci`

This displays a simple list of devices, for example:

```
~]$ lspci
00:00.0 Host bridge: Intel Corporation 82X38/X48 Express DRAM Controller
00:01.0 PCI bridge: Intel Corporation 82X38/X48 Express Host-Primary PCI Express Bridge
00:1a.0 USB Controller: Intel Corporation 82801I (ICH9 Family) USB UHCI Controller #4 (rev 02)
```

```
00:1a.1 USB Controller: Intel Corporation 82801I (ICH9 Family) USB UHCI Controller #5 (rev 02)
00:1a.2 USB Controller: Intel Corporation 82801I (ICH9 Family) USB UHCI Controller #6 (rev 02)
[output truncated]
```

You can also use the **-v** command-line option to display more verbose output, or **-vv** for very verbose output:

lspci -v|vv

For instance, to determine the manufacturer, model, and memory size of a system's video card, type:

```
~]$ lspci -v
[output truncated]
```

```
01:00.0 VGA compatible controller: nVidia Corporation G84 [Quadro FX 370] (rev a1) (prog-if 00
[VGA controller])
    Subsystem: nVidia Corporation Device 0491
    Physical Slot: 2
    Flags: bus master, fast devsel, latency 0, IRQ 16
    Memory at f2000000 (32-bit, non-prefetchable) [size=16M]
    Memory at e0000000 (64-bit, prefetchable) [size=256M]
    Memory at f0000000 (64-bit, non-prefetchable) [size=32M]
    I/O ports at 1100 [size=128]
    Expansion ROM at <unassigned> [disabled]
    Capabilities: <access denied>
    Kernel driver in use: nouveau
    Kernel modules: nouveau, nvidiafb
```

[output truncated]

For a complete list of available command-line options, see the **lspci(8)** manual page.

24.5.2. Using the lsusb Command

The **lsusb** command allows you to display information about USB buses and devices that are attached to them. To list all USB devices that are in the system, type the following at a shell prompt:

lsusb

This displays a simple list of devices, for example:

```
~]$ lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
[output truncated]
Bus 001 Device 002: ID 0bda:0151 Realtek Semiconductor Corp. Mass Storage Device (Multicard
Reader)
Bus 008 Device 002: ID 03f0:2c24 Hewlett-Packard Logitech M-UAL-96 Mouse
Bus 008 Device 003: ID 04b3:3025 IBM Corp.
```

You can also use the **-v** command-line option to display more verbose output:

lsusb -v

For instance:

```
~]$ lsusb -v
[output truncated]

Bus 008 Device 002: ID 03f0:2c24 Hewlett-Packard Logitech M-UAL-96 Mouse
Device Descriptor:
bLength          18
bDescriptorType   1
bcdUSB          2.00
bDeviceClass      0 (Defined at Interface level)
bDeviceSubClass    0
bDeviceProtocol    0
bMaxPacketSize0     8
idVendor        0x03f0 Hewlett-Packard
idProduct        0x2c24 Logitech M-UAL-96 Mouse
bcdDevice        31.00
iManufacturer      1
iProduct          2
iSerial           0
bNumConfigurations 1
Configuration Descriptor:
  bLength          9
  bDescriptorType   2
[output truncated]
```

For a complete list of available command-line options, see the **lsusb(8)** manual page.

24.5.3. Using the **lspcmcia** Command

The **lspcmcia** command allows you to list all PCMCIA devices that are present in the system. To do so, type the following at a shell prompt:

lspcmcia

For example:

```
~]$ lspcmcia
Socket 0 Bridge: [yenta_cardbus]      (bus ID: 0000:15:00.0)
```

You can also use the **-v** command-line option to display more verbose information, or **-vv** to increase the verbosity level even further:

lspcmcia -v|-vv

For instance:

```
~]$ lspcmcia -v
Socket 0 Bridge: [yenta_cardbus]      (bus ID: 0000:15:00.0)
  Configuration: state: on    ready: unknown
```

For a complete list of available command-line options, see the **pccardctl(8)** manual page.

24.5.4. Using the lscpu Command

The **lscpu** command allows you to list information about CPUs that are present in the system, including the number of CPUs, their architecture, vendor, family, model, CPU caches, etc. To do so, type the following at a shell prompt:

lscpu

For example:

```
~]$ lscpu
Architecture:      x86_64
CPU op-mode(s):    32-bit, 64-bit
Byte Order:        Little Endian
CPU(s):           4
On-line CPU(s) list: 0-3
Thread(s) per core: 1
Core(s) per socket: 4
Socket(s):        1
NUMA node(s):     1
Vendor ID:        GenuineIntel
CPU family:       6
Model:            23
Stepping:         7
CPU MHz:          1998.000
BogoMIPS:         4999.98
Virtualization:   VT-x
L1d cache:        32K
L1i cache:        32K
L2 cache:         3072K
NUMA node0 CPU(s): 0-3
```

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

24.6. MONITORING PERFORMANCE WITH NET-SNMP

Red Hat Enterprise Linux 6 includes the **Net-SNMP** software suite, which includes a flexible and extensible *Simple Network Management Protocol* (SNMP) agent. This agent and its associated utilities can be used to provide performance data from a large number of systems to a variety of tools which support polling over the SNMP protocol.

This section provides information on configuring the Net-SNMP agent to securely provide performance data over the network, retrieving the data using the SNMP protocol, and extending the SNMP agent to provide custom performance metrics.

24.6.1. Installing Net-SNMP

The Net-SNMP software suite is available as a set of RPM packages in the Red Hat Enterprise Linux software distribution. [Table 24.2, “Available Net-SNMP packages”](#) summarizes each of the packages and their contents.

Table 24.2. Available Net-SNMP packages

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

To install any of these packages, use the **yum** command in the following form:

yum install package

For example, to install the SNMP Agent Daemon and SNMP clients used in the rest of this section, type the following at a shell prompt:

~]# yum install net-snmp net-snmp-libs net-snmp-utils

Note that you must have superuser privileges (that is, you must be logged in as **root**) to run this command. For more information on how to install new packages in Red Hat Enterprise Linux, see [Section 8.2.4, “Installing Packages”](#).

24.6.2. Running the Net-SNMP Daemon

The **net-snmp** package contains **snmpd**, the SNMP Agent Daemon. This section provides information on how to start, stop, and restart the **snmpd** service, and shows how to enable it in a particular runlevel. For more information on the concept of runlevels and how to manage system services in Red Hat Enterprise Linux in general, see [Chapter 12, Services and Daemons](#).

24.6.2.1. Starting the Service

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

service snmpd start

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

chkconfig snmpd on

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

24.6.2.2. Stopping the Service

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

```
service snmpd stop
```

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

```
chkconfig snmpd off
```

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

24.6.2.3. Restarting the Service

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

```
service snmpd restart
```

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

```
service snmpd reload
```

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

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

24.6.3. Configuring Net-SNMP

To change the Net-SNMP Agent Daemon configuration, edit the **/etc/snmp/snmpd.conf** configuration file. The default **snmpd.conf** file shipped with Red Hat Enterprise Linux 6 is heavily commented and serves as a good starting point for agent configuration.

This section focuses on two common tasks: setting system information and configuring authentication. For more information about available configuration directives, see the **snmpd.conf(5)** manual page. Additionally, there is a utility in the net-snmp package named **snmpconf** which can be used to interactively generate a valid agent configuration.

Note that the net-snmp-utils package must be installed in order to use the **snmpwalk** utility described in this section.



NOTE

For any changes to the configuration file to take effect, force the **snmpd** service to re-read the configuration by running the following command as **root**:

```
service snmpd reload
```

24.6.3.1. Setting System Information

Net-SNMP provides some rudimentary system information via the **system** tree. For example, the following **snmpwalk** command shows the **system** tree with a default agent configuration.

```
~]# snmpwalk -v2c -c public localhost system
SNMPv2-MIB::sysDescr.0 = STRING: Linux localhost.localdomain 2.6.32-122.el6.x86_64 #1 SMP
Wed Mar 9 23:54:34 EST 2011 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (99554) 0:16:35.54
SNMPv2-MIB::sysContact.0 = STRING: Root <root@localhost> (configure
/etc/snmp/snmp.local.conf)
SNMPv2-MIB::sysName.0 = STRING: localhost.localdomain
SNMPv2-MIB::sysLocation.0 = STRING: Unknown (edit /etc/snmp/snmpd.conf)
```

By default, the **sysName** object is set to the host name. The **sysLocation** and **sysContact** objects can be configured in the **/etc/snmp/snmpd.conf** file by changing the value of the **syslocation** and **syscontact** directives, for example:

```
syslocation Datacenter, Row 3, Rack 2
syscontact UNIX Admin <admin@example.com>
```

After making changes to the configuration file, reload the configuration and test it by running the **snmpwalk** command again:

```
~]# service snmpd reload
Reloading snmpd: [ OK ]
~]# snmpwalk -v2c -c public localhost system
SNMPv2-MIB::sysDescr.0 = STRING: Linux localhost.localdomain 2.6.32-122.el6.x86_64 #1 SMP
Wed Mar 9 23:54:34 EST 2011 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (158357) 0:26:23.57
SNMPv2-MIB::sysContact.0 = STRING: UNIX Admin <admin@example.com>
SNMPv2-MIB::sysName.0 = STRING: localhost.localdomain
SNMPv2-MIB::sysLocation.0 = STRING: Datacenter, Row 3, Rack 2
```

24.6.3.2. Configuring Authentication

The Net-SNMP Agent Daemon supports all three versions of the SNMP protocol. The first two versions (1 and 2c) provide for simple authentication using a *community string*. This string is a shared secret between the agent and any client utilities. The string is passed in clear text over the network however and is not considered secure. Version 3 of the SNMP protocol supports user authentication and message encryption using a variety of protocols. The Net-SNMP agent also supports tunneling over SSH, TLS authentication with X.509 certificates, and Kerberos authentication.

Configuring SNMP Version 2c Community

To configure an **SNMP version 2c community**, use either the **rocommunity** or **rwcommunity** directive in the **/etc/snmp/snmpd.conf** configuration file. The format of the directives is the following:

```
directive community [source [OID]]
```

where *community* is the community string to use, *source* is an IP address or subnet, and *O/D* is the SNMP tree to provide access to. For example, the following directive provides read-only access to the **system** tree to a client using the community string “redhat” on the local machine:

```
rocommunity redhat 127.0.0.1 .1.3.6.1.2.1.1
```

To test the configuration, use the **snmpwalk** command with the **-v** and **-c** options.

```
~]# snmpwalk -v2c -c redhat localhost system
SNMPv2-MIB::sysDescr.0 = STRING: Linux localhost.localdomain 2.6.32-122.el6.x86_64 #1 SMP
Wed Mar 9 23:54:34 EST 2011 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (158357) 0:26:23.57
SNMPv2-MIB::sysContact.0 = STRING: UNIX Admin <admin@example.com>
SNMPv2-MIB::sysName.0 = STRING: localhost.localdomain
SNMPv2-MIB::sysLocation.0 = STRING: Datacenter, Row 3, Rack 2
```

Configuring SNMP Version 3 User

To configure an **SNMP version 3 user**, use the **net-snmp-create-v3-user** command. This command adds entries to the **/var/lib/net-snmp/snmpd.conf** and **/etc/snmp/snmpd.conf** files which create the user and grant access to the user. Note that the **net-snmp-create-v3-user** command may only be run when the agent is not running. The following example creates the “admin” user with the password “redhatsnmp”:

```
~]# service snmpd stop
Stopping snmpd: [ OK ]
~]# net-snmp-create-v3-user
Enter a SNMPv3 user name to create:
admin
Enter authentication pass-phrase:
redhatsnmp
Enter encryption pass-phrase:
[press return to reuse the authentication pass-phrase]

adding the following line to /var/lib/net-snmp/snmpd.conf:
createUser admin MD5 "redhatsnmp" DES
adding the following line to /etc/snmp/snmpd.conf:
rwuser admin
~]# service snmpd start
Starting snmpd: [ OK ]
```

The **rwuser** directive (or **rouser** when the **-ro** command-line option is supplied) that **net-snmp-create-v3-user** adds to **/etc/snmp/snmpd.conf** has a similar format to the **rwcommunity** and **rocommunity** directives:

directive user [noauth|auth|priv] [OID]

where *user* is a user name and *OID* is the SNMP tree to provide access to. By default, the Net-SNMP Agent Daemon allows only authenticated requests (the **auth** option). The **noauth** option allows you to permit unauthenticated requests, and the **priv** option enforces the use of encryption. The **authpriv** option specifies that requests must be authenticated and replies should be encrypted.

For example, the following line grants the user “admin” read-write access to the entire tree:

```
rwuser admin authpriv .1
```

To test the configuration, create a **.snmp** directory in your user’s home directory and a configuration file named **snmp.conf** in that directory (**~/.snmp/snmp.conf**) with the following lines:

```
defVersion 3
defSecurityLevel authPriv
defSecurityName admin
```

```
defPassphrase redhatsnmp
```

The **snmpwalk** command will now use these authentication settings when querying the agent:

```
~]$ snmpwalk -v3 localhost system
SNMPv2-MIB::sysDescr.0 = STRING: Linux localhost.localdomain 2.6.32-122.el6.x86_64 #1 SMP
Wed Mar 9 23:54:34 EST 2011 x86_64
[output truncated]
```

24.6.4. Retrieving Performance Data over SNMP

The Net-SNMP Agent in Red Hat Enterprise Linux provides a wide variety of performance information over the SNMP protocol. In addition, the agent can be queried for a listing of the installed RPM packages on the system, a listing of currently running processes on the system, or the network configuration of the system.

This section provides an overview of OIDs related to performance tuning available over SNMP. It assumes that the `net-snmp-utils` package is installed and that the user is granted access to the SNMP tree as described in [Section 24.6.3.2, “Configuring Authentication”](#).

24.6.4.1. Hardware Configuration

The **Host Resources MIB** included with Net-SNMP presents information about the current hardware and software configuration of a host to a client utility. [Table 24.3, “Available OIDs”](#) summarizes the different OIDs available under that MIB.

Table 24.3. Available OIDs

OID	Description
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.

24.6.4.2. CPU and Memory Information

Most system performance data is available in the **UCD SNMP MIB**. The **systemStats** OID provides a number of counters around processor usage:

```
~]$ snmpwalk localhost UCD-SNMP-MIB::systemStats
UCD-SNMP-MIB::sslIndex.0 = INTEGER: 1
UCD-SNMP-MIB::ssErrorName.0 = STRING: systemStats
UCD-SNMP-MIB::ssSwapIn.0 = INTEGER: 0 kB
UCD-SNMP-MIB::ssSwapOut.0 = INTEGER: 0 kB
UCD-SNMP-MIB::ssIOSent.0 = INTEGER: 0 blocks/s
UCD-SNMP-MIB::ssIOReceive.0 = INTEGER: 0 blocks/s
UCD-SNMP-MIB::ssSysInterrupts.0 = INTEGER: 29 interrupts/s
UCD-SNMP-MIB::ssSysContext.0 = INTEGER: 18 switches/s
UCD-SNMP-MIB::ssCpuUser.0 = INTEGER: 0
UCD-SNMP-MIB::ssCpuSystem.0 = INTEGER: 0
UCD-SNMP-MIB::ssCpuIdle.0 = INTEGER: 99
UCD-SNMP-MIB::ssCpuRawUser.0 = Counter32: 2278
UCD-SNMP-MIB::ssCpuRawNice.0 = Counter32: 1395
UCD-SNMP-MIB::ssCpuRawSystem.0 = Counter32: 6826
UCD-SNMP-MIB::ssCpuRawIdle.0 = Counter32: 3383736
UCD-SNMP-MIB::ssCpuRawWait.0 = Counter32: 7629
UCD-SNMP-MIB::ssCpuRawKernel.0 = Counter32: 0
UCD-SNMP-MIB::ssCpuRawInterrupt.0 = Counter32: 434
UCD-SNMP-MIB::ssIORawSent.0 = Counter32: 266770
UCD-SNMP-MIB::ssIORawReceived.0 = Counter32: 427302
UCD-SNMP-MIB::ssRawInterrupts.0 = Counter32: 743442
UCD-SNMP-MIB::ssRawContexts.0 = Counter32: 718557
UCD-SNMP-MIB::ssCpuRawSoftIRQ.0 = Counter32: 128
UCD-SNMP-MIB::ssRawSwapIn.0 = Counter32: 0
UCD-SNMP-MIB::ssRawSwapOut.0 = Counter32: 0
```

In particular, the **ssCpuRawUser**, **ssCpuRawSystem**, **ssCpuRawWait**, and **ssCpuRawIdle** OIDs provide counters which are helpful when determining whether a system is spending most of its processor time in kernel space, user space, or I/O. **ssRawSwapIn** and **ssRawSwapOut** can be helpful when determining whether a system is suffering from memory exhaustion.

More memory information is available under the **UCD-SNMP-MIB::memory** OID, which provides similar data to the **free** command:

```
~]$ snmpwalk localhost UCD-SNMP-MIB::memory
UCD-SNMP-MIB::memIndex.0 = INTEGER: 0
```

```
UCD-SNMP-MIB::memErrorName.0 = STRING: swap
UCD-SNMP-MIB::memTotalSwap.0 = INTEGER: 1023992 kB
UCD-SNMP-MIB::memAvailSwap.0 = INTEGER: 1023992 kB
UCD-SNMP-MIB::memTotalReal.0 = INTEGER: 1021588 kB
UCD-SNMP-MIB::memAvailReal.0 = INTEGER: 634260 kB
UCD-SNMP-MIB::memTotalFree.0 = INTEGER: 1658252 kB
UCD-SNMP-MIB::memMinimumSwap.0 = INTEGER: 16000 kB
UCD-SNMP-MIB::memBuffer.0 = INTEGER: 30760 kB
UCD-SNMP-MIB::memCached.0 = INTEGER: 216200 kB
UCD-SNMP-MIB::memSwapError.0 = INTEGER: noError(0)
UCD-SNMP-MIB::memSwapErrorMsg.0 = STRING:
```

Load averages are also available in the **UCD SNMP MIB**. The SNMP table **UCD-SNMP-MIB::laTable** has a listing of the 1, 5, and 15 minute load averages:

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

laIndex laNames laLoad laConfig laLoadInt laLoadFloat laErrorFlag laErrMessage
 1 Load-1 0.00 12.00      0 0.000000  noError
 2 Load-5 0.00 12.00      0 0.000000  noError
 3 Load-15 0.00 12.00     0 0.000000  noError
```

24.6.4.3. File System and Disk Information

The **Host Resources MIB** provides information on file system size and usage. Each file system (and also each memory pool) has an entry in the **HOST-RESOURCES-MIB::hrStorageTable** table:

```
~]$ snmpstable -Cb localhost HOST-RESOURCES-MIB::hrStorageTable
SNMP table: HOST-RESOURCES-MIB::hrStorageTable

Index          Type      Descr
AllocationUnits  Size  Used AllocationFailures
 1    HOST-RESOURCES-TYPES::hrStorageRam Physical memory
1024 Bytes 1021588 388064      ?
 3 HOST-RESOURCES-TYPES::hrStorageVirtualMemory Virtual memory
1024 Bytes 2045580 388064      ?
 6    HOST-RESOURCES-TYPES::hrStorageOther Memory buffers
1024 Bytes 1021588 31048      ?
 7    HOST-RESOURCES-TYPES::hrStorageOther Cached memory
1024 Bytes 216604 216604      ?
 10 HOST-RESOURCES-TYPES::hrStorageVirtualMemory   Swap space
1024 Bytes 1023992 0      ?
 31   HOST-RESOURCES-TYPES::hrStorageFixedDisk      /
4096 Bytes 2277614 250391      ?
 35   HOST-RESOURCES-TYPES::hrStorageFixedDisk   /dev/shm
4096 Bytes 127698 0      ?
 36   HOST-RESOURCES-TYPES::hrStorageFixedDisk   /boot
1024 Bytes 198337 26694      ?
```

The OIDs under **HOST-RESOURCES-MIB::hrStorageSize** and **HOST-RESOURCES-MIB::hrStorageUsed** can be used to calculate the remaining capacity of each mounted file system.

I/O data is available both in **UCD-SNMP-MIB::systemStats** (**ssIORawSent.0** and **ssIORawReceived.0**) and in **UCD-DISKIO-MIB::diskIOTable**. The latter provides much more granular data. Under this table

are OIDs for **diskIONReadX** and **diskIONWrittenX**, which provide counters for the number of bytes read from and written to the block device in question since the system boot:

```
~]$ snmpstable -Cb localhost UCD-DISKIO-MIB::diskIOTable
SNMP table: UCD-DISKIO-MIB::diskIOTable

Index Device NRead NWritten Reads Writes LA1 LA5 LA15 NReadX NWrittenX
...
25 sda 216886272 139109376 16409 4894 ? ? ? 216886272 139109376
26 sda1 2455552 5120 613 2 ? ? ? 2455552 5120
27 sda2 1486848 0 332 0 ? ? ? 1486848 0
28 sda3 212321280 139104256 15312 4871 ? ? ? 212321280 139104256
```

24.6.4.4. Network Information

The **Interfaces MIB** provides information on network devices. **IF-MIB::ifTable** provides an SNMP table with an entry for each interface on the system, the configuration of the interface, and various packet counters for the interface. The following example shows the first few columns of **ifTable** on a system with two physical network interfaces:

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

Index Descr Type Mtu Speed PhysAddress AdminStatus
1 lo softwareLoopback 16436 100000000 up
2 eth0 ethernetCsmacd 1500 0 52:54:0:c7:69:58 up
3 eth1 ethernetCsmacd 1500 0 52:54:0:a7:a3:24 down
```

Network traffic is available under the OIDs **IF-MIB::ifOutOctets** and **IF-MIB::ifInOctets**. The following SNMP queries will retrieve network traffic for each of the interfaces on this system:

```
~]$ snmpwalk localhost IF-MIB::ifDescr
IF-MIB::ifDescr.1 = STRING: lo
IF-MIB::ifDescr.2 = STRING: eth0
IF-MIB::ifDescr.3 = STRING: eth1
~]$ snmpwalk localhost IF-MIB::ifOutOctets
IF-MIB::ifOutOctets.1 = Counter32: 10060699
IF-MIB::ifOutOctets.2 = Counter32: 650
IF-MIB::ifOutOctets.3 = Counter32: 0
~]$ snmpwalk localhost IF-MIB::ifInOctets
IF-MIB::ifInOctets.1 = Counter32: 10060699
IF-MIB::ifInOctets.2 = Counter32: 78650
IF-MIB::ifInOctets.3 = Counter32: 0
```

24.6.5. Extending Net-SNMP

The Net-SNMP Agent can be extended to provide application metrics in addition to raw system metrics. This allows for capacity planning as well as performance issue troubleshooting. For example, it may be helpful to know that an email system had a 5-minute load average of 15 while being tested, but it is more helpful to know that the email system has a load average of 15 while processing 80,000 messages a second. When application metrics are available via the same interface as the system metrics, this also allows for the visualization of the impact of different load scenarios on system performance (for example, each additional 10,000 messages increases the load average linearly until 100,000).

A number of the applications that ship with Red Hat Enterprise Linux extend the Net-SNMP Agent to provide application metrics over SNMP. There are several ways to extend the agent for custom applications as well. This section describes extending the agent with shell scripts and Perl plug-ins. It assumes that the `net-snmp-utils` and `net-snmp-perl` packages are installed, and that the user is granted access to the SNMP tree as described in [Section 24.6.3.2, “Configuring Authentication”](#).

24.6.5.1. Extending Net-SNMP with Shell Scripts

The Net-SNMP Agent provides an extension MIB (**NET-SNMP-EXTEND-MIB**) that can be used to query arbitrary shell scripts. To specify the shell script to run, use the **extend** directive in the `/etc/snmp/snmpd.conf` file. Once defined, the Agent will provide the exit code and any output of the command over SNMP. The example below demonstrates this mechanism with a script which determines the number of **httpd** processes in the process table.



NOTE

The Net-SNMP Agent also provides a built-in mechanism for checking the process table via the **proc** directive. See the `snmpd.conf(5)` manual page for more information.

The exit code of the following shell script is the number of **httpd** processes running on the system at a given point in time:

```
#!/bin/sh

NUMPIDS=`pgrep httpd | wc -l`

exit $NUMPIDS
```

To make this script available over SNMP, copy the script to a location on the system path, set the executable bit, and add an **extend** directive to the `/etc/snmp/snmpd.conf` file. The format of the **extend** directive is the following:

```
extend name prog args
```

where *name* is an identifying string for the extension, *prog* is the program to run, and *args* are the arguments to give the program. For instance, if the above shell script is copied to `/usr/local/bin/check_apache.sh`, the following directive will add the script to the SNMP tree:

```
extend httpd_pids /bin/sh /usr/local/bin/check_apache.sh
```

The script can then be queried at **NET-SNMP-EXTEND-MIB::nsExtendObjects**:

```
~]$ snmpwalk localhost NET-SNMP-EXTEND-MIB::nsExtendObjects
NET-SNMP-EXTEND-MIB::nsExtendNumEntries.0 = INTEGER: 1
NET-SNMP-EXTEND-MIB::nsExtendCommand."httpd_pids" = STRING: /bin/sh
NET-SNMP-EXTEND-MIB::nsExtendArgs."httpd_pids" = STRING: /usr/local/bin/check_apache.sh
NET-SNMP-EXTEND-MIB::nsExtendInput."httpd_pids" = STRING:
NET-SNMP-EXTEND-MIB::nsExtendCacheTime."httpd_pids" = INTEGER: 5
NET-SNMP-EXTEND-MIB::nsExtendExecType."httpd_pids" = INTEGER: exec(1)
NET-SNMP-EXTEND-MIB::nsExtendRunType."httpd_pids" = INTEGER: run-on-read(1)
NET-SNMP-EXTEND-MIB::nsExtendStorage."httpd_pids" = INTEGER: permanent(4)
NET-SNMP-EXTEND-MIB::nsExtendStatus."httpd_pids" = INTEGER: active(1)
NET-SNMP-EXTEND-MIB::nsExtendOutput1Line."httpd_pids" = STRING:
```

```
NET-SNMP-EXTEND-MIB::nsExtendOutputFull."httpd_pids" = STRING:  
NET-SNMP-EXTEND-MIB::nsExtendOutNumLines."httpd_pids" = INTEGER: 1  
NET-SNMP-EXTEND-MIB::nsExtendResult."httpd_pids" = INTEGER: 8  
NET-SNMP-EXTEND-MIB::nsExtendOutLine."httpd_pids".1 = STRING:
```

Note that the exit code ("8" in this example) is provided as an INTEGER type and any output is provided as a STRING type. To expose multiple metrics as integers, supply different arguments to the script using the **extend** directive. For example, the following shell script can be used to determine the number of processes matching an arbitrary string, and will also output a text string giving the number of processes:

```
#!/bin/sh

PATTERN=$1
NUMPIDS=`pgrep $PATTERN | wc -l`

echo "There are $NUMPIDS $PATTERN processes."
exit $NUMPIDS
```

The following **/etc/snmp/snmpd.conf** directives will give both the number of **httpd** PIDs as well as the number of **snmpd** PIDs when the above script is copied to **/usr/local/bin/check_proc.sh**:

```
extend httpd_pids /bin/sh /usr/local/bin/check_proc.sh httpd
extend snmpd_pids /bin/sh /usr/local/bin/check_proc.sh snmpd
```

The following example shows the output of an **snmpwalk** of the **nsExtendObjects** OID:

```
~]$ snmpwalk localhost NET-SNMP-EXTEND-MIB::nsExtendObjects
NET-SNMP-EXTEND-MIB::nsExtendNumEntries.0 = INTEGER: 2
NET-SNMP-EXTEND-MIB::nsExtendCommand."httpd_pids" = STRING: /bin/sh
NET-SNMP-EXTEND-MIB::nsExtendCommand."snmpd_pids" = STRING: /bin/sh
NET-SNMP-EXTEND-MIB::nsExtendArgs."httpd_pids" = STRING: /usr/local/bin/check_proc.sh httpd
NET-SNMP-EXTEND-MIB::nsExtendArgs."snmpd_pids" = STRING: /usr/local/bin/check_proc.sh
snmpd
NET-SNMP-EXTEND-MIB::nsExtendInput."httpd_pids" = STRING:
NET-SNMP-EXTEND-MIB::nsExtendInput."snmpd_pids" = STRING:
...
NET-SNMP-EXTEND-MIB::nsExtendResult."httpd_pids" = INTEGER: 8
NET-SNMP-EXTEND-MIB::nsExtendResult."snmpd_pids" = INTEGER: 1
NET-SNMP-EXTEND-MIB::nsExtendOutLine."httpd_pids".1 = STRING: There are 8 httpd processes.
NET-SNMP-EXTEND-MIB::nsExtendOutLine."snmpd_pids".1 = STRING: There are 1 snmpd
processes.
```



WARNING

Integer exit codes are limited to a range of 0–255. For values that are likely to exceed 256, either use the standard output of the script (which will be typed as a string) or a different method of extending the agent.

This last example shows a query for the free memory of the system and the number of **httpd** processes. This query could be used during a performance test to determine the impact of the number of processes on memory pressure:

```
~]$ snmpget localhost \
'NET-SNMP-EXTEND-MIB::nsExtendResult."httpd_pids"' \
UCD-SNMP-MIB::memAvailReal.0
NET-SNMP-EXTEND-MIB::nsExtendResult."httpd_pids" = INTEGER: 8
UCD-SNMP-MIB::memAvailReal.0 = INTEGER: 799664 kB
```

24.6.5.2. Extending Net-SNMP with Perl

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

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

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

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

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

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



NOTE

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

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

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

The handler function should respond to two types of SNMP requests: the GET request and the GETNEXT request. The type of request is determined by calling the **getMode** method on the **request_info** object passed as the third parameter to the handler function. If the request is a GET

request, the caller will expect the handler to set the value of the **request** object, depending on the OID of the request. If the request is a GETNEXT request, the caller will also expect the handler to set the OID of the request to the next available OID in the tree. This is illustrated in the following code example:

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

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

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

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

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

```
#!/usr/bin/perl

use NetSNMP::agent (':all');
use NetSNMP::ASN qw(ASN_OCTET_STR ASN_INTEGER);

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

    for($request = $requests; $request; $request = $request->next()) {
        my $oid = $request->getOID();
        if ($request_info->getMode() == MODE_GET) {
            if ($oid == new NetSNMP::OID(".1.3.6.1.4.1.8072.9999.9999.1.0")) {
```

```

        $request->setValue(ASN_OCTET_STR, $string_value);
    }
} elseif ($oid == new NetSNMP::OID(".1.3.6.1.4.1.8072.9999.9999.1.1")) {
    $request->setValue(ASN_INTEGER, $integer_value);
}
} elseif ($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, $integer_value);
    }
    elseif ($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::netSnmpPlaypen
NET-SNMP-MIB::netSnmpPlaypen.1.0 = STRING: "hello world"
NET-SNMP-MIB::netSnmpPlaypen.1.1 = INTEGER: 8675309
```

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

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

24.7. ADDITIONAL RESOURCES

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

24.7.1. Installed Documentation

- **ps(1)** – The manual page for the **ps** command.
- **top(1)** – The manual page for the **top** command.
- **free(1)** – The manual page for the **free** command.

- **df(1)** – The manual page for the **df** command.
- **du(1)** – The manual page for the **du** command.
- **lspci(8)** – The manual page for the **lspci** command.
- **snmpd(8)** – The manual page for the **snmpd** service.
- **snmpd.conf(5)** – The manual page for the **/etc/snmp/snmpd.conf** file containing full documentation of available configuration directives.

CHAPTER 25. VIEWING AND MANAGING LOG FILES

Log files are files that contain messages about the system, including the kernel, services, and applications running on it. There are different log files for different information. For example, there is a default system log file, a log file just for security messages, and a log file for cron tasks.

Log files can be very useful when trying to troubleshoot a problem with the system such as trying to load a kernel driver or when looking for unauthorized login attempts to the system. This chapter discusses where to find log files, how to view log files, and what to look for in log files.

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

25.1. INSTALLING RSYSLOG

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

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

25.1.1. Upgrading to rsyslog version 7

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

Procedure 25.1. Upgrading to rsyslog 7

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

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

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

The yum shell prompt appears.

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

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

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

```
> run
--> Running transaction check
```

```

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

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

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

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

```

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

```

Finished Transaction
> quit
Leaving Shell
~]#

```

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

25.2. LOCATING LOG FILES

A list of log files maintained by **rsyslogd** can be found in the **/etc/rsyslog.conf** configuration file. Most log files are located in the **/var/log/** directory. Some applications such as **httpd** and **samba** have a directory within **/var/log/** for their log files.

You may notice multiple files in the **/var/log/** directory with numbers after them (for example, **cron-20100906**). These numbers represent a time stamp that has been added to a rotated log file. Log files are rotated so their file sizes do not become too large. The **logrotate** package contains a cron task that automatically rotates log files according to the **/etc/logrotate.conf** configuration file and the configuration files in the **/etc/logrotate.d/** directory.

25.3. BASIC CONFIGURATION OF RSYSLOG

The main configuration file for **rsyslog** is **/etc/rsyslog.conf**. Here, you can specify *global directives*, *modules*, and *rules* that consist of *filter* and *action* parts. Also, you can add comments in the form of text following a hash sign (#).

25.3.1. Filters

A rule is specified by a *filter* part, which selects a subset of syslog messages, and an *action* part, which specifies what to do with the selected messages. To define a rule in your **/etc/rsyslog.conf** configuration file, define both, a filter and an action, on one line and separate them with one or more spaces or tabs.

rsyslog offers various ways to filter syslog messages according to selected properties. The available filtering methods can be divided into *Facility/Priority-based*, *Property-based*, and *Expression-based* filters.

Facility/Priority-based filters

The most used and well-known way to filter syslog messages is to use the facility/priority-based filters which filter syslog messages based on two conditions: *facility* and *priority* separated by a dot. To create a selector, use the following syntax:

FACILITY.PRIORITY

where:

- **FACILITY** specifies the subsystem that produces a specific syslog message. For example, the **mail** subsystem handles all mail-related syslog messages. **FACILITY** can be represented by one of the following keywords (or by a numerical code): **kern** (0), **user** (1), **mail** (2), **daemon** (3), **auth** (4), **syslog** (5), **lpr** (6), **news** (7), **uucp** (8), **cron** (9), **authpriv** (10), **ftp** (11), and **local0** through **local7** (16 - 23).
- **PRIORITY** specifies a priority of a syslog message. **PRIORITY** can be represented by one of the following keywords (or by a number): **debug** (7), **info** (6), **notice** (5), **warning** (4), **err** (3), **crit** (2), **alert** (1), and **emerg** (0).

The aforementioned syntax selects syslog messages with the defined or *higher* priority. By preceding any priority keyword with an equal sign (=), you specify that only syslog messages with the specified priority will be selected. All other priorities will be ignored. Conversely, preceding a priority keyword with an exclamation mark (!) selects all syslog messages except those with the defined priority.

In addition to the keywords specified above, you may also use an asterisk (*) to define all facilities or priorities (depending on where you place the asterisk, before or after the comma). Specifying the priority keyword **none** serves for facilities with no given priorities. Both facility and priority conditions are case-insensitive.

To define multiple facilities and priorities, separate them with a comma (,). To define multiple selectors on one line, separate them with a semi-colon (;). Note that each selector in the selector field is capable of overwriting the preceding ones, which can exclude some priorities from the pattern.

Example 25.1. Facility/Priority-based Filters

The following are a few examples of simple facility/priority-based filters that can be specified in **/etc/rsyslog.conf**. To select all kernel syslog messages with any priority, add the following text into the configuration file:

kern.*

To select all mail syslog messages with priority **crit** and higher, use this form:

mail.crit

To select all cron syslog messages except those with the **info** or **debug** priority, set the configuration in the following form:

cron.!info,!debug

Property-based filters

Property-based filters let you filter syslog messages by any property, such as **timegenerated** or **syslogtag**. For more information on properties, see [the section called “Properties”](#). You can compare each of the specified properties to a particular value using one of the compare-operations listed in [Table 25.1, “Property-based compare-operations”](#). Both property names and compare operations are case-sensitive.

Property-based filter must start with a colon (:). To define the filter, use the following syntax:

:*PROPERTY*, [!]*COMPARE_OPERATION*, "STRING"

where:

- The *PROPERTY* attribute specifies the desired property.
- The optional exclamation point (!) negates the output of the compare-operation. Other Boolean operators are currently not supported in property-based filters.
- The *COMPARE_OPERATION* attribute specifies one of the compare-operations listed in [Table 25.1, “Property-based compare-operations”](#).
- The *STRING* attribute specifies the value that the text provided by the property is compared to. This value must be enclosed in quotation marks. To escape certain character inside the string (for example a quotation mark (")), use the backslash character (\).

Table 25.1. Property-based compare-operations

Compare-operation	Description
contains	Checks whether the provided string matches any part of the text provided by the property. To perform case-insensitive comparisons, use contains_i .
isequal	Compares the provided string against all of the text provided by the property. These two values must be exactly equal to match.
startswith	Checks whether the provided string is found exactly at the beginning of the text provided by the property. To perform case-insensitive comparisons, use startswith_i .
regex	Compares the provided POSIX BRE (Basic Regular Expression) against the text provided by the property.

Compare-operation	Description
ereregex	Compares the provided POSIX ERE (Extended Regular Expression) regular expression against the text provided by the property.
isempty	Checks if the property is empty. The value is discarded. This is especially useful when working with normalized data, where some fields may be populated based on normalization result.

Example 25.2. Property-based Filters

The following are a few examples of property-based filters that can be specified in **/etc/rsyslog.conf**. To select syslog messages which contain the string **error** in their message text, use:

:msg, contains, "error"

The following filter selects syslog messages received from the host name **host1**:

:hostname, isEqual, "host1"

To select syslog messages which do not contain any mention of the words **fatal** and **error** with any or no text between them (for example, **fatal lib error**), type:

:msg, !regex, "fatal .* error"

Expression-based filters

Expression-based filters select syslog messages according to defined arithmetic, Boolean or string operations. Expression-based filters use **rsyslog**'s own scripting language called *RainerScript* to build complex filters.

The basic syntax of expression-based filter looks as follows:

if *EXPRESSION* then *ACTION* else *ACTION*

where:

- The *EXPRESSION* attribute represents an expression to be evaluated, for example: **\$msg startsWith 'DEVNAME'** or **\$syslogfacility-text == 'local0'**. You can specify more than one expression in a single filter by using **and** and **or** operators.
- The *ACTION* attribute represents an action to be performed if the expression returns the value **true**. This can be a single action, or an arbitrary complex script enclosed in curly braces.
- Expression-based filters are indicated by the keyword **if** at the start of a new line. The **then** keyword separates the *EXPRESSION* from the *ACTION*. Optionally, you can employ the **else** keyword to specify what action is to be performed in case the condition is not met.

With expression-based filters, you can nest the conditions by using a script enclosed in curly braces

as in [Example 25.3, "Expression-based Filters"](#). The script allows you to use *facility/priority-based* filters inside the expression. On the other hand, *property-based* filters are not recommended here. RainerScript supports regular expressions with specialized functions **re_match()** and **re_extract()**.

Example 25.3. Expression-based Filters

The following expression contains two nested conditions. The log files created by a program called *prog1* are split into two files based on the presence of the "test" string in the message.

```
if $programname == 'prog1' then {
    action(type="omfile" file="/var/log/prog1.log")
    if $msg contains 'test' then
        action(type="omfile" file="/var/log/prog1test.log")
    else
        action(type="omfile" file="/var/log/prog1notest.log")
}
```

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

25.3.2. Actions

Actions specify what is to be done with the messages filtered out by an already-defined selector. The following are some of the actions you can define in your rule:

Saving syslog messages to log files

The majority of actions specify to which log file a syslog message is saved. This is done by specifying a file path after your already-defined selector:

FILTER PATH

where *FILTER* stands for user-specified selector and *PATH* is a path of a target file.

For instance, the following rule is comprised of a selector that selects all **cron** syslog messages and an action that saves them into the **/var/log/cron.log** log file:

cron.* /var/log/cron.log

By default, the log file is synchronized every time a syslog message is generated. Use a dash mark (-) as a prefix of the file path you specified to omit syncing:

FILTER -PATH

Note that you might lose information if the system terminates right after a write attempt. However, this setting can improve performance, especially if you run programs that produce very verbose log messages.

Your specified file path can be either *static* or *dynamic*. Static files are represented by a fixed file path as shown in the example above. Dynamic file paths can differ according to the received message. Dynamic file paths are represented by a template and a question mark (?) prefix:

FILTER ?DynamicFile

where *DynamicFile* is a name of a predefined template that modifies output paths. You can use the dash prefix (-) to disable syncing, also you can use multiple templates separated by a colon (;). For more information on templates, see [the section called "Generating Dynamic File Names"](#).

If the file you specified is an existing **terminal** or **/dev/console** device, syslog messages are sent to standard output (using special **terminal**-handling) or your console (using special **/dev/console**-handling) when using the X Window System, respectively.

Sending syslog messages over the network

rsyslog allows you to send and receive syslog messages over the network. This feature allows you to administer syslog messages of multiple hosts on one machine. To forward syslog messages to a remote machine, use the following syntax:

```
@[zNUMBER]HOST:[PORT]
```

where:

- The at sign (@) indicates that the syslog messages are forwarded to a host using the **UDP** protocol. To use the **TCP** protocol, use two at signs with no space between them (@@).
- The optional **zNUMBER** setting enables **zlib** compression for syslog messages. The **NUMBER** attribute specifies the level of compression (from 1 – lowest to 9 – maximum). Compression gain is automatically checked by **rsyslogd**, messages are compressed only if there is any compression gain and messages below 60 bytes are never compressed.
- The **HOST** attribute specifies the host which receives the selected syslog messages.
- The **PORT** attribute specifies the host machine's port.

When specifying an **IPv6** address as the host, enclose the address in square brackets ([]).

Example 25.4. Sending syslog Messages over the Network

The following are some examples of actions that forward syslog messages over the network (note that all actions are preceded with a selector that selects all messages with any priority). To forward messages to **192.168.0.1** via the **UDP** protocol, type:

```
*.* @192.168.0.1
```

To forward messages to "example.com" using port 6514 and the **TCP** protocol, use:

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

The following compresses messages with **zlib** (level 9 compression) and forwards them to **2001:db8::1** using the **UDP** protocol

```
*.* @(z9)[2001:db8::1]
```

Output channels

Output channels are primarily used to specify the maximum size a log file can grow to. This is very

useful for log file rotation (for more information see [Section 25.3.5, "Log Rotation"](#)). An output channel is basically a collection of information about the output action. Output channels are defined by the **\$outchannel** directive. To define an output channel in **/etc/rsyslog.conf**, use the following syntax:

```
$outchannel NAME, FILE_NAME, MAX_SIZE, ACTION
```

where:

- The *NAME* attribute specifies the name of the output channel.
- The *FILE_NAME* attribute specifies the name of the output file. Output channels can write only into files, not pipes, terminal, or other kind of output.
- The *MAX_SIZE* attribute represents the maximum size the specified file (in *FILE_NAME*) can grow to. This value is specified in bytes.
- The *ACTION* attribute specifies the action that is taken when the maximum size, defined in *MAX_SIZE*, is hit.

To use the defined output channel as an action inside a rule, type:

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

Example 25.5. Output channel log rotation

The following output shows a simple log rotation through the use of an output channel. First, the output channel is defined via the **\$outchannel** directive:

```
$outchannel log_rotation, /var/log/test_log.log, 104857600, /home/joe/log_rotation_script
```

and then it is used in a rule that selects every syslog message with any priority and executes the previously-defined output channel on the acquired syslog messages:

```
*.* :omfile:$log_rotation
```

Once the limit (in the example **100 MB**) is hit, the **/home/joe/log_rotation_script** is executed. This script can contain anything from moving the file into a different folder, editing specific content out of it, or simply removing it.

Sending syslog messages to specific users

rsyslog can send syslog messages to specific users by specifying a user name of the user you want to send the messages to (as in [Example 25.7, "Specifying Multiple Actions"](#)). To specify more than one user, separate each user name with a comma (,). To send messages to every user that is currently logged on, use an asterisk (*).

Executing a program

rsyslog lets you execute a program for selected syslog messages and uses the **system()** call to execute the program in shell. To specify a program to be executed, prefix it with a caret character (^). Consequently, specify a template that formats the received message and passes it to the specified executable as a one line parameter (for more information on templates, see [Section 25.3.3, "Templates"](#)).

FILTER ^EXECUTABLE; TEMPLATE

Here an output of the *FILTER* condition is processed by a program represented by *EXECUTABLE*. This program can be any valid executable. Replace *TEMPLATE* with the name of the formatting template.

Example 25.6. Executing a Program

In the following example, any syslog message with any priority is selected, formatted with the **template** template and passed as a parameter to the **test-program** program, which is then executed with the provided parameter:

```
*.* ^test-program;template
```

WARNING

When accepting messages from any host, and using the shell execute action, you may be vulnerable to command injection. An attacker may try to inject and execute commands in the program you specified to be executed in your action. To avoid any possible security threats, thoroughly consider the use of the shell execute action.

Storing syslog messages in a database

Selected syslog messages can be directly written into a database table using the *database writer* action. The database writer uses the following syntax:

```
:PLUGIN:DB_HOST,DB_NAME,DB_USER,DB_PASSWORD:[TEMPLATE]
```

where:

- The *PLUGIN* calls the specified plug-in that handles the database writing (for example, the **ommysql** plug-in).
- The *DB_HOST* attribute specifies the database host name.
- The *DB_NAME* attribute specifies the name of the database.
- The *DB_USER* attribute specifies the database user.
- The *DB_PASSWORD* attribute specifies the password used with the aforementioned database user.
- The *TEMPLATE* attribute specifies an optional use of a template that modifies the syslog message. For more information on templates, see [Section 25.3.3, “Templates”](#).



IMPORTANT

Currently, **rsyslog** provides support for **MySQL** and **PostgreSQL** databases only. In order to use the **MySQL** and **PostgreSQL** database writer functionality, install the `rsyslog-mysql` and `rsyslog-pgsql` packages, respectively. Also, make sure you load the appropriate modules in your `/etc/rsyslog.conf` configuration file:

```
$ModLoad ommysql # Output module for MySQL support
$ModLoad ompgsql # Output module for PostgreSQL support
```

For more information on **rsyslog** modules, see [Section 25.7, “Using Rsyslog Modules”](#).

Alternatively, you may use a generic database interface provided by the **omlibdb** module (supports: Firebird/Interbase, MS SQL, Sybase, SQLLite, Ingres, Oracle, mSQL).

Discarding syslog messages

To discard your selected messages, use the tilde character (~).

FILTER ~

The discard action is mostly used to filter out messages before carrying on any further processing. It can be effective if you want to omit some repeating messages that would otherwise fill the log files. The results of discard action depend on where in the configuration file it is specified, for the best results place these actions on top of the actions list. Please note that once a message has been discarded there is no way to retrieve it in later configuration file lines.

For instance, the following rule discards any cron syslog messages:

 cron.* ~

Specifying Multiple Actions

For each selector, you are allowed to specify multiple actions. To specify multiple actions for one selector, write each action on a separate line and precede it with an ampersand (&) character:

FILTER ACTION
 & *ACTION*
 & *ACTION*

Specifying multiple actions improves the overall performance of the desired outcome since the specified selector has to be evaluated only once.

Example 25.7. Specifying Multiple Actions

In the following example, all kernel syslog messages with the critical priority (**crit**) are sent to user **user1**, processed by the template **temp** and passed on to the **test-program** executable, and forwarded to **192.168.0.1** via the **UDP** protocol.

```
kern.=crit user1
& ^test-program;temp
& @192.168.0.1
```

Any action can be followed by a template that formats the message. To specify a template, suffix an action with a semicolon (;) and specify the name of the template. For more information on templates, see [Section 25.3.3, "Templates"](#).



WARNING

A template must be defined before it is used in an action, otherwise it is ignored. In other words, template definitions should always precede rule definitions in `/etc/rsyslog.conf`.

25.3.3. Templates

Any output that is generated by `rsyslog` can be modified and formatted according to your needs with the use of *templates*. To create a template use the following syntax in `/etc/rsyslog.conf`:

```
$template TEMPLATE_NAME,"text %PROPERTY% more text", [OPTION]
```

where:

- **\$template** is the template directive that indicates that the text following it, defines a template.
- **TEMPLATE_NAME** is the name of the template. Use this name to refer to the template.
- Anything between the two quotation marks ("...") is the actual template text. Within this text, special characters, such as `\n` for new line or `\r` for carriage return, can be used. Other characters, such as `%` or `"`, have to be escaped if you want to use those characters literally.
- The text specified between two percent signs (`%`) specifies a *property* that allows you to access specific contents of a syslog message. For more information on properties, see [the section called "Properties"](#).
- The **OPTION** attribute specifies any options that modify the template functionality. The currently supported template options are **sql** and **stdsql**, which are used for formatting the text as an SQL query.



NOTE

Note that the database writer checks whether the **sql** or **stdsql** options are specified in the template. If they are not, the database writer does not perform any action. This is to prevent any possible security threats, such as SQL injection.

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

Generating Dynamic File Names

Templates can be used to generate dynamic file names. By specifying a property as a part of the file path, a new file will be created for each unique property, which is a convenient way to classify syslog messages.

For example, use the ***timegenerated*** property, which extracts a time stamp from the message, to generate a unique file name for each syslog message:

```
$template DynamicFile,"/var/log/test_logs/%timegenerated%-test.log"
```

Keep in mind that the ***\$template*** directive only specifies the template. You must use it inside a rule for it to take effect. In **/etc/rsyslog.conf**, use the question mark (?) in an action definition to mark the dynamic file name template:

```
*.* ?DynamicFile
```

Properties

Properties defined inside a template (between two percent signs (%)) enable access various contents of a syslog message through the use of a *property replacer*. To define a property inside a template (between the two quotation marks ("...")), use the following syntax:

```
%PROPERTY_NAME[:FROM_CHAR:TO_CHAR:OPTION]%
```

where:

- The *PROPERTY_NAME* attribute specifies the name of a property. A list of all available properties and their detailed description can be found in the **rsyslog.conf(5)** manual page under the section *Available Properties*.
- *FROM_CHAR* and *TO_CHAR* attributes denote a range of characters that the specified property will act upon. Alternatively, regular expressions can be used to specify a range of characters. To do so, set the letter **R** as the *FROM_CHAR* attribute and specify your desired regular expression as the *TO_CHAR* attribute.
- The *OPTION* attribute specifies any property options, such as the **lowercase** option to convert the input to lowercase. A list of all available property options and their detailed description can be found in the **rsyslog.conf(5)** manual page under the section *Property Options*.

The following are some examples of simple properties:

- The following property obtains the whole message text of a syslog message:

```
%msg%
```

- The following property obtains the first two characters of the message text of a syslog message:

```
%msg:1:2%
```

- The following property obtains the whole message text of a syslog message and drops its last line feed character:

```
%msg:::drop-last-lf%
```

- The following property obtains the first 10 characters of the time stamp that is generated when the syslog message is received and formats it according to the [RFC 3999](#) date standard.

```
%timegenerated:1:10:date-rfc3339%
```

Template Examples

This section presents a few examples of **rsyslog** templates.

[Example 25.8, "A verbose syslog message template"](#) shows a template that formats a syslog message so that it outputs the message's severity, facility, the time stamp of when the message was received, the host name, the message tag, the message text, and ends with a new line.

Example 25.8. A verbose syslog message template

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

[Example 25.9, "A wall message template"](#) shows a template that resembles a traditional wall message (a message that is send to every user that is logged in and has their **mesg(1)** permission set to **yes**). This template outputs the message text, along with a host name, message tag and a time stamp, on a new line (using **\r** and **\n**) and rings the bell (using **\7**).

Example 25.9. A wall message template

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

[Example 25.10, "A database formatted message template"](#) shows a template that formats a syslog message so that it can be used as a database query. Notice the use of the **sql** option at the end of the template specified as the template option. It tells the database writer to format the message as an MySQL **SQL** query.

Example 25.10. A database formatted message template

```
$template dbFormat,"insert into SystemEvents (Message, Facility, FromHost, Priority,
DeviceReportedTime, ReceivedAt, InfoUnitID, SysLogTag) values ('%msg%', %syslogfacility%,
'%HOSTNAME%', %syslogpriority%, '%timereported:::date-mysql%', '%timegenerated:::date-
mysql%', %iut%, '%syslogtag%')", sql
```

rsyslog also contains a set of predefined templates identified by the **RSYSLOG_** prefix. These are reserved for the syslog's use and it is advisable to not create a template using this prefix to avoid conflicts. The following list shows these predefined templates along with their definitions.

RSYSLOG_DebugFormat

A special format used for troubleshooting property problems.

```
"Debug line with all properties:\\nFROMHOST: '%FROMHOST%', fromhost-ip: '%fromhost-ip%',
HOSTNAME: '%HOSTNAME%', PRI: %PRI%,\\nsyslogtag '%syslogtag%', programname:
'%programname%', APP-NAME: '%APP-NAME%', PROCID: '%PROCID%', MSGID:
'%MSGID%',\\nTIMESTAMP: '%TIMESTAMP%', STRUCTURED-DATA: '%STRUCTURED-
DATA%',\\nmsg: '%msg%'\\nescaped msg: '%msg:::drop-cc%'\\nrawmsg: '%rawmsg%'\\n\\n'"
```

RSYSLOG_SyslogProtocol23Format

The format specified in IETF's internet-draft ietf-syslog-protocol-23, which is assumed to become the new syslog standard RFC.

```
"%PRI%1 %TIMESTAMP:::date-rfc3339% %HOSTNAME% %APP-NAME% %PROCID%
%MSGID% %STRUCTURED-DATA% %msg%\n\"
```

RSYSLOG_FileFormat

A modern-style logfile format similar to TraditionalFileFormat, but with high-precision time stamps and time zone information.

```
"%TIMESTAMP:::date-rfc3339% %HOSTNAME% %syslogtag%%msg:::sp-if-no-1st-
sp%%msg:::drop-last-lf%\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-
lf%\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\"
```

25.3.4. Global Directives

Global directives are configuration options that apply to the **rsyslogd** daemon. They usually specify a value for a specific predefined variable that affects the behavior of the **rsyslogd** daemon or a rule that follows. All of the global directives must start with a dollar sign (\$). Only one directive can be specified per line. The following is an example of a global directive that specifies the maximum size of the syslog message queue:

```
$MainMsgQueueSize 50000
```

The default size defined for this directive (**10,000** messages) can be overridden by specifying a different value (as shown in the example above).

You can define multiple directives in your **/etc/rsyslog.conf** configuration file. A directive affects the behavior of all configuration options until another occurrence of that same directive is detected. Global directives can be used to configure actions, queues and for debugging. A comprehensive list of all available configuration directives can be found in [the section called “Online Documentation”](#). Currently, a new configuration format has been developed that replaces the \$-based syntax (see [Section 25.4, “Using the New Configuration Format”](#)). However, classic global directives remain supported as a legacy format.

25.3.5. Log Rotation

The following is a sample **/etc/logrotate.conf** configuration file:

```
# rotate log files weekly
weekly
# keep 4 weeks worth of backlogs
rotate 4
# uncomment this if you want your log files compressed
compress
```

All of the lines in the sample configuration file define global options that apply to every log file. In our example, log files are rotated weekly, rotated log files are kept for four weeks, and all rotated log files are compressed by **gzip** into the **.gz** format. Any lines that begin with a hash sign (#) are comments and are not processed.

You may define configuration options for a specific log file and place it under the global options. However, it is advisable to create a separate configuration file for any specific log file in the **/etc/logrotate.d/** directory and define any configuration options there.

The following is an example of a configuration file placed in the **/etc/logrotate.d/** directory:

```
/var/log/messages {
    rotate 5
    weekly
    postrotate
        /usr/bin/killall -HUP 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.

25.4. USING THE NEW CONFIGURATION FORMAT

In **rsyslog** version 7, available for Red Hat Enterprise Linux 6 in the **rsyslog7** package, a new configuration syntax is introduced. This new configuration format aims to be more powerful, more intuitive, and to prevent common mistakes by not permitting certain invalid constructs. The syntax enhancement is enabled by the new configuration processor that relies on RainerScript. The legacy format is still fully supported and it is used by default in the **/etc/rsyslog.conf** configuration file. To install rsyslog 7, see [Section 25.1.1, “Upgrading to rsyslog version 7”](#).

RainerScript is a scripting language designed for processing network events and configuring event processors such as **rsyslog**. The version of RainerScript in rsyslog version 5 is used to define expression-based filters, see [Example 25.3, “Expression-based Filters”](#). The version of RainerScript in rsyslog version 7 implements the **input()** and **ruleset()** statements, which permit the **/etc/rsyslog.conf** configuration file to be written in the new syntax. The new syntax differs mainly in that it is much more structured; parameters are passed as arguments to statements, such as input, action, template, and module load. The scope of options is limited by blocks. This enhances readability and reduces the number of bugs caused by misconfiguration. There is also a significant performance gain. Some functionality is exposed in both syntaxes, some only in the new one.

Compare the configuration written with legacy-style parameters:

```
$InputFileName /tmp/inputfile
$inputFileTag tag1:
$inputFileStateFile inputfile-state
$inputRunFileMonitor
```

and the same configuration with the use of the new format statement:

```
input(type="imfile" file="/tmp/inputfile" tag="tag1:" statefile="inputfile-state")
```

This significantly reduces the number of parameters used in configuration, improves readability, and also provides higher execution speed. For more information on RainerScript statements and parameters see [the section called “Online Documentation”](#).

25.4.1. Rulesets

Leaving special directives aside, **rsyslog** handles messages as defined by **rules** that consist of a filter condition and an action to be performed if the condition is true. With a traditionally written

/etc/rsyslog.conf file, all rules are evaluated in order of appearance for every input message. This process starts with the first rule and continues until all rules have been processed or until the message is discarded by one of the rules.

However, rules can be grouped into sequences called *rulesets*. With rulesets, you can limit the effect of certain rules only to selected inputs or enhance the performance of **rsyslog** by defining a distinct set of actions bound to a specific input. In other words, filter conditions that will be inevitably evaluated as false for certain types of messages can be skipped. The legacy ruleset definition in **/etc/rsyslog.conf** can look as follows:

```
$RuleSet rulesetname
rule
rule2
```

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

```
$RuleSet RSYSLOG_DefaultRuleset
```

With the new configuration format in rsyslog 7, the **input()** and **ruleset()** statements are reserved for this operation. The new format ruleset definition in **/etc/rsyslog.conf** can look as follows:

```
ruleset(name="rulesetname") {
    rule
    rule2
    call rulesetname2
    ...
}
```

Replace *rulesetname* with an identifier for your ruleset. The ruleset name cannot start with **RSYSLOG_** since this namespace is reserved for use by **rsyslog**. **RSYSLOG_DefaultRuleset** then defines the default set of rules to be performed if the message has no other ruleset assigned. With *rule* and *rule2* you can define rules in filter-action format mentioned above. With the **call** parameter, you can nest rulesets by calling them from inside other ruleset blocks.

After creating a ruleset, you need to specify what input it will apply to:

```
input(type="input_type" port="port_num" ruleset="rulesetname");
```

Here you can identify an input message by *input_type*, which is an input module that gathered the message, or by *port_num* – the port number. Other parameters such as *file* or *tag* can be specified for **input()**. Replace *rulesetname* with a name of the ruleset to be evaluated against the message. In case an input message is not explicitly bound to a ruleset, the default ruleset is triggered.

You can also use the legacy format to define rulesets, for more information see [the section called "Online Documentation"](#).

Example 25.11. Using rulesets

The following rulesets ensure different handling of remote messages coming from different ports. Add the following into **/etc/rsyslog.conf**:

```
ruleset(name="remote-6514") {
    action(type="omfile" file="/var/log/remote-6514")
}
```

```

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

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

```

Rulesets shown in the above example define log destinations for the remote input from two ports, in case of port **601**, messages are sorted according to the facility. Then, the TCP input is enabled and bound to rulesets. Note that you must load the required modules (imtcp) for this configuration to work.

25.4.2. Compatibility with sysklogd

The compatibility mode specified via the **-c** option exists in **rsyslog** version 5 but not in version 7. Also, the sysklogd-style command-line options are deprecated and configuring **rsyslog** through these command-line options should be avoided. However, you can use several templates and directives to configure **rsyslogd** to emulate sysklogd-like behavior.

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

25.5. WORKING WITH QUEUES IN RSYSLOG

Queues are used to pass content, mostly syslog messages, between components of **rsyslog**. With queues, rsyslog is capable of processing multiple messages simultaneously and to apply several actions to a single message at once. The data flow inside **rsyslog** can be illustrated as follows:

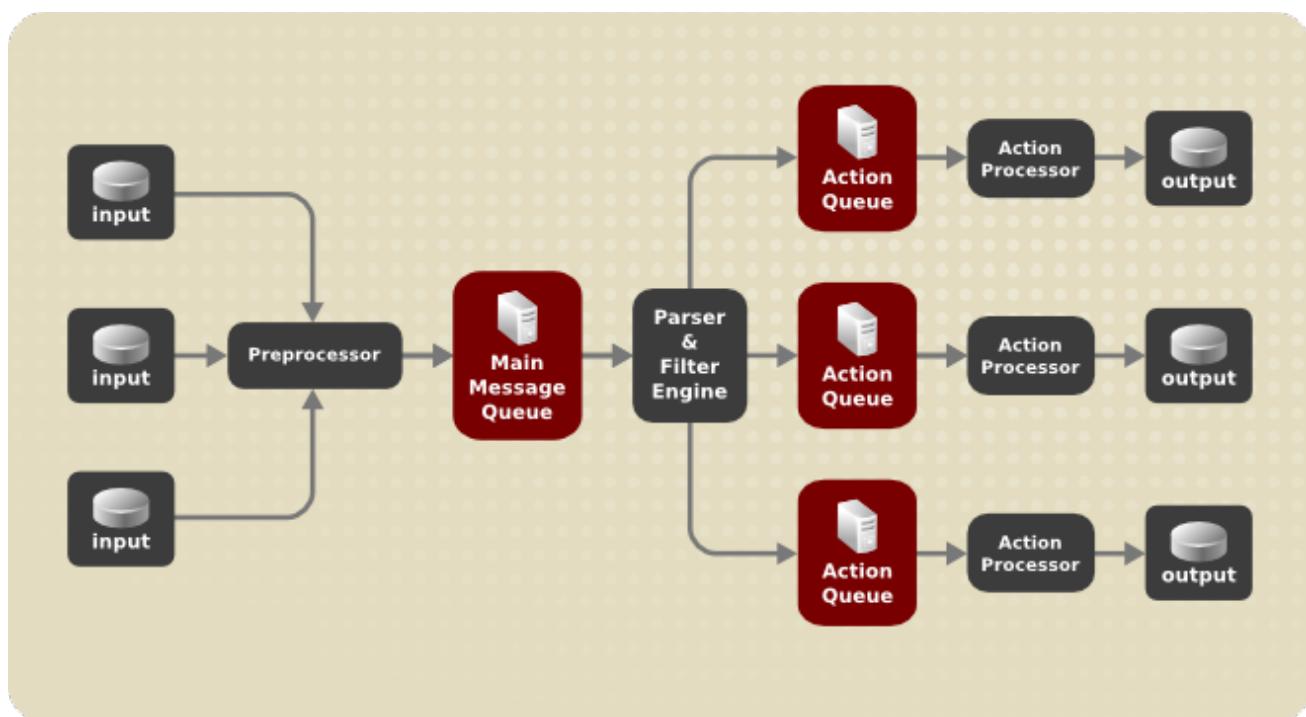


Figure 25.1. Message Flow in Rsyslog

Whenever **rsyslog** receives a message, it passes this message to the preprocessor and then places it into the *main message queue*. Messages wait there to be dequeued and passed to the *rule processor*.

The *rule processor* is a parsing and filtering engine. Here, the rules defined in `/etc/rsyslog.conf` are applied. Based on these rules, the rule processor evaluates which actions are to be performed. Each action has its own action queue. Messages are passed through this queue to the respective action processor which creates the final output. Note that at this point, several actions can run simultaneously on one message. For this purpose, a message is duplicated and passed to multiple action processors.

Only one queue per action is possible. Depending on configuration, the messages can be sent right to the action processor without action queuing. This is the behavior of *direct queues* (see below). In case the output action fails, the action processor notifies the action queue, which then takes an unprocessed element back and after some time interval, the action is attempted again.

To sum up, there are two positions where queues stand in **rsyslog**: either in front of the rule processor as a single *main message queue* or in front of various types of output actions as *action queues*. Queues provide two main advantages that both lead to increased performance of message processing:

- they serve as buffers that *decouple* producers and consumers in the structure of **rsyslog**
- they allow for *parallelization* of actions performed on messages

Apart from this, queues can be configured with several directives to provide optimal performance for your system. These configuration options are covered in the following sections.



WARNING

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

25.5.1. Defining Queues

Based on where the messages are stored, there are several types of queues: *direct*, *in-memory*, *disk*, and *disk-assisted in-memory* queues that are most widely used. You can choose one of these types for the main message queue and also for action queues. Add the following into `/etc/rsyslog.conf`:

```
$objectQueueType queue_type
```

Here, you can apply the setting for the main message queue (replace *object* with **MainMsg**) or for an action queue (replace *object* with **Action**). Replace *queue_type* with one of **direct**, **linkedlist** or **fixedarray** (which are in-memory queues), or **disk**.

The default setting for a main message queue is the **FixedArray** queue with a limit of 10,000 messages. Action queues are by default set as Direct queues.

Direct Queues

For many simple operations, such as when writing output to a local file, building a queue in front of an action is not needed. To avoid queuing, use:



`$objectQueueType Direct`

Replace *object* with **MainMsg** or with **Action** to use this option to the main message queue or for an action queue respectively. With direct queue, messages are passed directly and immediately from the producer to the consumer.

Disk Queues

Disk queues store messages strictly on a hard drive, which makes them highly reliable but also the slowest of all possible queuing modes. This mode can be used to prevent the loss of highly important log data. However, disk queues are not recommended in most use cases. To set a disk queue, type the following into **/etc/rsyslog.conf**:

`$objectQueueType Disk`

Replace *object* with **MainMsg** or with **Action** to use this option to the main message queue or for an action queue respectively. Disk queues are written in parts, with a default size 10 Mb. This default size can be modified with the following configuration directive:

`$objectQueueMaxFileSize size`

where *size* represents the specified size of disk queue part. The defined size limit is not restrictive, **rsyslog** always writes one complete queue entry, even if it violates the size limit. Each part of a disk queue matches with an individual file. The naming directive for these files looks as follows:

`$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 25.12. Reliable Forwarding of Log Messages to a Server

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

Procedure 25.2. Forwarding To a Single Server

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

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

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

Where:

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

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

Procedure 25.3. Forwarding To Multiple Servers

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

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

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

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

25.5.2. Creating a New Directory for rsyslog Log Files

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

Procedure 25.4. Creating a New Working Directory

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

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

2. Install utilities to manage SELinux policy:

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

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

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

4. Apply the SELinux context:

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

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

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

6. Create subdirectories as required. For example:

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

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

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

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

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

25.5.3. Managing Queues

All types of queues can be further configured to match your requirements. You can use several directives to modify both action queues and the main message queue. Currently, there are more than 20 queue parameters available, see [the section called “Online Documentation”](#). Some of these settings are used commonly, others, such as worker thread management, provide closer control over the queue behavior and are reserved for advanced users. With advanced settings, you can optimize **rsyslog**’s performance, schedule queuing, or modify the behavior of a queue on system shutdown.

Limiting Queue Size

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

```
$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 Dequeuing

To increase performance, you can configure **rsyslog** to dequeue multiple messages at once. To set the upper limit for such dequeuing, use:

```
$objectQueueDequeueBatchSize number
```

Replace *number* with 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.

25.5.4. Using the New Syntax for rsyslog queues

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

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

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

Example 25.13. Defining an Action Queue

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

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

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

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

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

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

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

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

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

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

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

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

Where:

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

25.6. CONFIGURING RSYSLOG ON A LOGGING SERVER

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1. ~]# semanage port -a -t syslogd_port_t -p tcp 10514
2. Review the SELinux ports by entering the following command:

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

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

```
~]# service rsyslog restart
```

4. Verify which ports **rsyslog** is now listening to:

```
~]# netstat -tnlp | grep rsyslog
tcp      0      0 0.0.0.0:10514          0.0.0.0:*    LISTEN    2528/rsyslogd
tcp      0      0 ::1:10514             ::*:      LISTEN    2528/rsyslogd
```

See the **semanage-port(8)** manual page for more information on the **semanage port** command.

Procedure 25.6. Configuring The iptables Firewall

Configure the **iptables** firewall to allow incoming **rsyslog** traffic. For example, to allow **TCP** traffic on port **10514**, proceed as follows:

1. Open the **/etc/sysconfig/iptables** file in a text editor.
2. Add an **INPUT** rule allowing **TCP** traffic on port **10514** to the file. The new rule must appear before any **INPUT** rules that **REJECT** traffic.

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 10514 -j ACCEPT
```

3. Save the changes to the **/etc/sysconfig/iptables** file.
4. Restart the **iptables** service for the firewall changes to take effect.

```
~]# service iptables restart
```

Procedure 25.7. Configuring rsyslog to Receive and Sort Remote Log Messages

1. Open the **/etc/rsyslog.conf** file in a text editor and proceed as follows:
 - a. Add these lines below the modules section but above the **Provides UDP syslog reception** section:

```
# Define templates before the rules that use them

### Per-Host Templates for Remote Systems ####
$template TmplAuthpriv,
"/var/log/remote/auth/%HOSTNAME%/%PROGRAMNAME:::secpath-replace%.log"
$template TmplMsg,
"/var/log/remote/msg/%HOSTNAME%/%PROGRAMNAME:::secpath-replace%.log"
```

- b. Replace the default **Provides TCP syslog reception** section with the following:

```
# Provides TCP syslog reception
$ModLoad imtcp
# Adding this ruleset to process remote messages
$RuleSet remote1
authpriv.* ?TmplAuthpriv
```

```
*.info;mail.none;authpriv.none;cron.none ?TmplMsg
$RuleSet RSYSLOG_DefaultRuleset #End the rule set by switching back to the default
rule set
$InputTCPServerBindRuleset remote1 #Define a new input and bind it to the "remote1"
rule set
$InputTCPServerRun 10514
```

Save the changes to the **/etc/rsyslog.conf** file.

2. The **rsyslog** service must be running on both the logging server and the systems attempting to log to it.

- a. Use the **service** command to start the **rsyslog** service.

```
~]# service rsyslog start
```

- b. To ensure the **rsyslog** service starts automatically in future, enter the following command as root:

```
~]# chkconfig rsyslog on
```

Your log server is now configured to receive and store log files from the other systems in your environment.

25.6.1. Using The New Template Syntax on a Logging Server

Rsyslog 7 has a number of different templates styles. The string template most closely resembles the legacy format. Reproducing the templates from the example above using the string format would look as follows:

```
template(name="TmplAuthpriv" type="string"
        string="/var/log/remote/auth/%HOSTNAME%/%PROGRAMNAME:::secpath-replace%.log"
      )
template(name="TmplMsg" 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="TmplAuthpriv" type="list") {
  constant(value="/var/log/remote/auth/")
  property(name="hostname")
  constant(value="/")
  property(name="programname" SecurePath="replace")
  constant(value=".log")
}

template(name="TmplMsg" 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="TmplAuthpriv")
    *.info;mail.none;authpriv.none;cron.none action(type="omfile" DynaFile="TmplMsg")
}

input(type="imtcp" port="10514" ruleset="remote1")
```

25.7. USING RSYSLOG MODULES

Due to its modular design, **rsyslog** offers a variety of *modules* which provide additional functionality. Note that modules can be written by third parties. Most modules provide additional inputs (see *Input Modules* below) or outputs (see *Output Modules* below). Other modules provide special functionality specific to each module. The modules may provide additional configuration directives that become available after a module is loaded. To load a module, use the following syntax:

\$ModLoad MODULE

where **\$ModLoad** is the global directive that loads the specified module and *MODULE* represents your desired module. For example, if you want to load the Text File Input Module (**imfile**) that enables **rsyslog** to convert any standard text files into syslog messages, specify the following line in the **/etc/rsyslog.conf** configuration file:

\$ModLoad imfile

rsyslog offers a number of modules which are split into the following main categories:

- Input Modules – Input modules gather messages from various sources. The name of an input module always starts with the **im** prefix, such as **imfile**.
- Output Modules – Output modules provide a facility to issue message to various targets such as sending across a network, storing in a database, or encrypting. The name of an output module always starts with the **om** prefix, such as **omsnmp**, **omrelp**, and so on.
- Parser Modules – These modules are useful in creating custom parsing rules or to parse malformed messages. With moderate knowledge of the C programming language, you can create your own message parser. The name of a parser module always starts with the **pm** prefix, such as **pmrfc5424**, **pmrfc3164**, and so on.
- Message Modification Modules – Message modification modules change content of syslog messages. Names of these modules start with the **mm** prefix. Message Modification Modules such as **mmanon**, **mmnormalize**, or **mmjsonparse** are used for anonymization or normalization of messages.

- String Generator Modules – String generator modules generate strings based on the message content and strongly cooperate with the template feature provided by **rsyslog**. For more information on templates, see [Section 25.3.3, "Templates"](#). The name of a string generator module always starts with the **sm** prefix, such as **smfile** or **smtradfile**.
- Library Modules – Library modules provide functionality for other loadable modules. These modules are loaded automatically by **rsyslog** when needed and cannot be configured by the user.

A comprehensive list of all available modules and their detailed description can be found at http://www.rsyslog.com/doc/rsyslog_conf_modules.html.



WARNING

Note that when **rsyslog** loads any modules, it provides them with access to some of its functions and data. This poses a possible security threat. To minimize security risks, use trustworthy modules only.

25.7.1. Importing Text Files

The Text File Input Module, abbreviated as **imfile**, enables **rsyslog** to convert any text file into a stream of syslog messages. You can use **imfile** to import log messages from applications that create their own text file logs. To load **imfile**, add the following into **/etc/rsyslog.conf**:

```
$ModLoad imfile
$InputFilePollInterval int
```

It is sufficient to load **imfile** once, even when importing multiple files. The **\$InputFilePollInterval** global directive specifies how often **rsyslog** checks for changes in connected text files. The default interval is 10 seconds, to change it, replace *int* with a time interval specified in seconds.

To identify the text files to import, use the following syntax in **/etc/rsyslog.conf**:

```
# File 1
$InputFileName path_to_file
$InputFileTag tag:
$InputFileStateFile state_file_name
$InputFileSeverity severity
$InputFileFacility facility
$InputRunFileMonitor

# File 2
$InputFileName path_to_file2
...
...
```

Four settings are required to specify an input text file:

- replace *path_to_file* with a path to the text file.
- replace *tag*: with a tag name for this message.

- replace `state_file_name` with a unique name for the `state file`. *State files*, which are stored in the `rsyslog` working directory, keep cursors for the monitored files, marking what partition has already been processed. If you delete them, whole files will be read in again. Make sure that you specify a name that does not already exist.
- add the `$InputRunFileMonitor` directive that enables the file monitoring. Without this setting, the text file will be ignored.

Apart from the required directives, there are several other settings that can be applied on the text input. Set the severity of imported messages by replacing `severity` with an appropriate keyword. Replace `facility` with a keyword to define the subsystem that produced the message. The keywords for severity and facility are the same as those used in facility/priority-based filters, see [Section 25.3.1, "Filters"](#).

Example 25.15. Importing Text Files

The Apache HTTP server creates log files in text format. To apply the processing capabilities of `rsyslog` to apache error messages, first use the `imfile` module to import the messages. Add the following into `/etc/rsyslog.conf`:

```
$ModLoad imfile
$InputFileName /var/log/httpd/error_log
$InputFileTag apache-error:
$InputFileStateFile state-apache-error
$InputRunFileMonitor
```

25.7.2. Exporting Messages to a Database

Processing of log data can be faster and more convenient when performed in a database rather than with text files. Based on the type of DBMS used, choose from various output modules such as `ommysql`, `ompgsql`, `omoracle`, or `ommongodb`. As an alternative, use the generic `omlibdbi` output module that relies on the `libdbi` library. The `omlibdbi` module supports database systems Firebird/Interbase, MS SQL, Sybase, SQLite, Ingres, Oracle, mSQL, MySQL, and PostgreSQL.

Example 25.16. Exporting Rsyslog Messages to a Database

To store the rsyslog messages in a MySQL database, add the following into `/etc/rsyslog.conf`:

```
$ModLoad ommysql
$actionOmmysqlServerPort 1234
*:.* :ommysql:database-server,database-name,database-userid,database-password
```

First, the output module is loaded, then the communication port is specified. Additional information, such as name of the server and the database, and authentication data, is specified on the last line of the above example.

25.7.3. Enabling Encrypted Transport

Confidentiality and integrity in network transmissions can be provided by either the `TLS` or `GSSAPI` encryption protocol.

Transport Layer Security (TLS) is a cryptographic protocol designed to provide communication security over the network. When using TLS, rsyslog messages are encrypted before sending, and mutual authentication exists between the sender and receiver.

Generic Security Service API (GSSAPI) is an application programming interface for programs to access security services. To use it in connection with **rsyslog** you must have a functioning **Kerberos** environment.

25.7.4. Using RELP

Reliable Event Logging Protocol (RELP) is a networking protocol for data logging in computer networks. It is designed to provide reliable delivery of event messages, which makes it useful in environments where message loss is not acceptable.

To configure RELP, first install the **rsyslog-relp** package both on the server and the client:

```
~]# yum install rsyslog-relp
```

Then, configure both the server and the client.

1. To configure the client, configure:
 - loading the required modules
 - the TCP input port
 - the transport settings

by adding the following configuration to the **/etc/rsyslog.conf** file:

```
$ModLoad omrelp
$ModLoad imuxsock
$ModLoad imtcp
$InputTCPServerRun "port"
.* :omrelp:"target_IP":"target_port"
```

Replace *port* to start a listener at the required port.

Replace *target_IP* and *target_port* with the IP address and port that identify the target server.

2. To configure the server:
 - configure loading the modules
 - configure the TCP input similarly to the client configuration
 - configure the rules and choose an action to be performed

by adding the following configuration to the **/etc/rsyslog.conf** file:

```
$ModLoad imuxsock
$ModLoad imrelp
$RuleSet relp
.* "log_path"
$InputRELPServerBindRuleset relp
$InputRELPServerRun "target_port"
```

■ Replace *target_port* with the same value as on the clients.

In the previous example, *log_path* specifies the path for storing messages.

25.8. DEBUGGING RSYSLOG

To run **rsyslogd** in debugging mode, use the following command:

```
■ rsyslogd -dn
```

With this command, **rsyslogd** produces debugging information and prints it to the standard output. The **-n** stands for "no fork". You can modify debugging with environmental variables, for example, you can store the debug output in a log file. Before starting **rsyslogd**, type the following on the command line:

```
■ export RSYSLOG_DEBUGLOG="path"  
■ export RSYSLOG_DEBUG="Debug"
```

Replace *path* with a desired location for the file where the debugging information will be logged. For a complete list of options available for the RSYSLOG_DEBUG variable, see the related section in the **rsyslogd(8)** manual page.

To check if syntax used in the **/etc/rsyslog.conf** file is valid use:

```
■ rsyslogd -N 1
```

Where **1** represents level of verbosity of the output message. This is a forward compatibility option because currently, only one level is provided. However, you must add this argument to run the validation.

25.9. MANAGING LOG FILES IN A GRAPHICAL ENVIRONMENT

As an alternative to the aforementioned command-line utilities, Red Hat Enterprise Linux 6 provides an accessible GUI for managing log messages.

25.9.1. Viewing Log Files

Most log files are stored in plain text format. You can view them with any text editor such as **Vi** or **Emacs**. Some log files are readable by all users on the system; however, root privileges are required to read most log files.

To view system log files in an interactive, real-time application, use the **Log File Viewer**.



NOTE

In order to use the **Log File Viewer**, first ensure the `gnome-system-log` package is installed on your system by running, as **root**:

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

The `gnome-system-log` package is provided by the Optional subscription channel that must be enabled before installation. See [Section 8.4.8, “Adding the Optional and Supplementary Repositories”](#) for more information on Red Hat additional channels. For more information on installing packages with Yum, see [Section 8.2.4, “Installing Packages”](#).

After you have installed the `gnome-system-log` package, open the **Log File Viewer** by clicking **Applications → System Tools → Log File Viewer**, or type the following command at a shell prompt:

```
~]$ gnome-system-log
```

The application only displays log files that exist; thus, the list might differ from the one shown in [Figure 25.2, “Log File Viewer”](#).

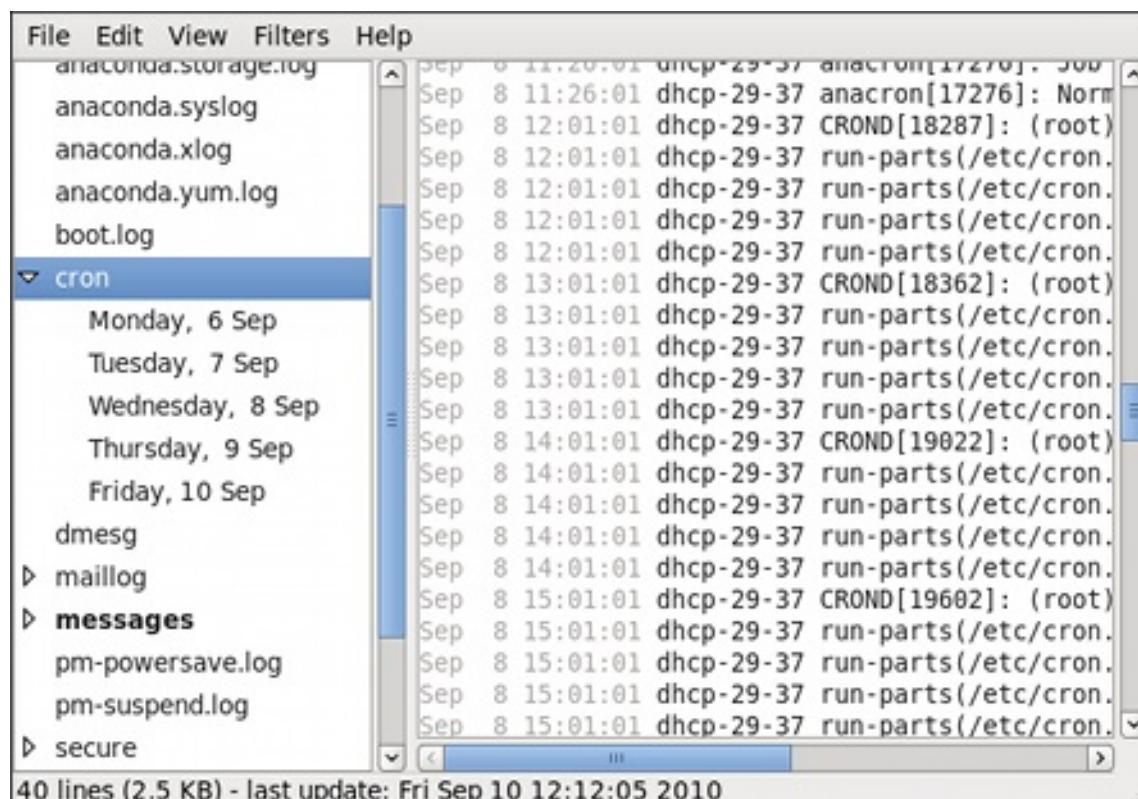


Figure 25.2. Log File Viewer

The **Log File Viewer** application lets you filter any existing log file. Click on **Filters** from the menu and select **Manage Filters** to define or edit the desired filter.

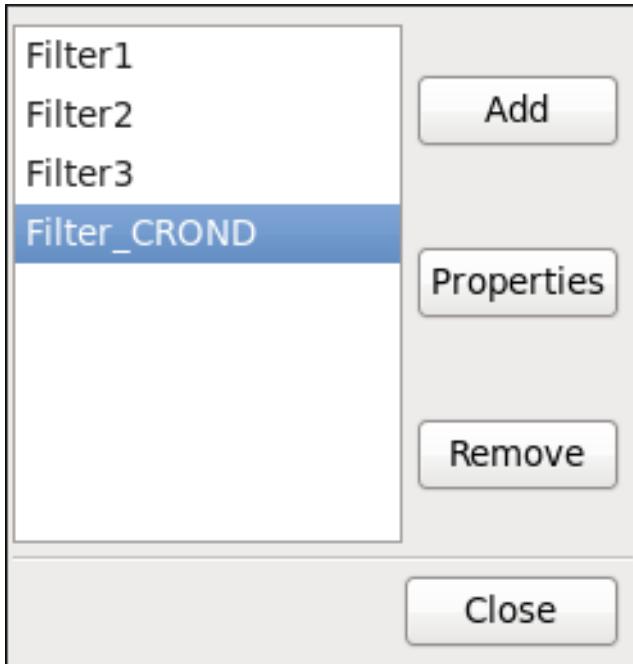


Figure 25.3. Log File Viewer - Filters

Adding or editing a filter lets you define its parameters as is shown in Figure 25.4, "Log File Viewer - defining a filter".

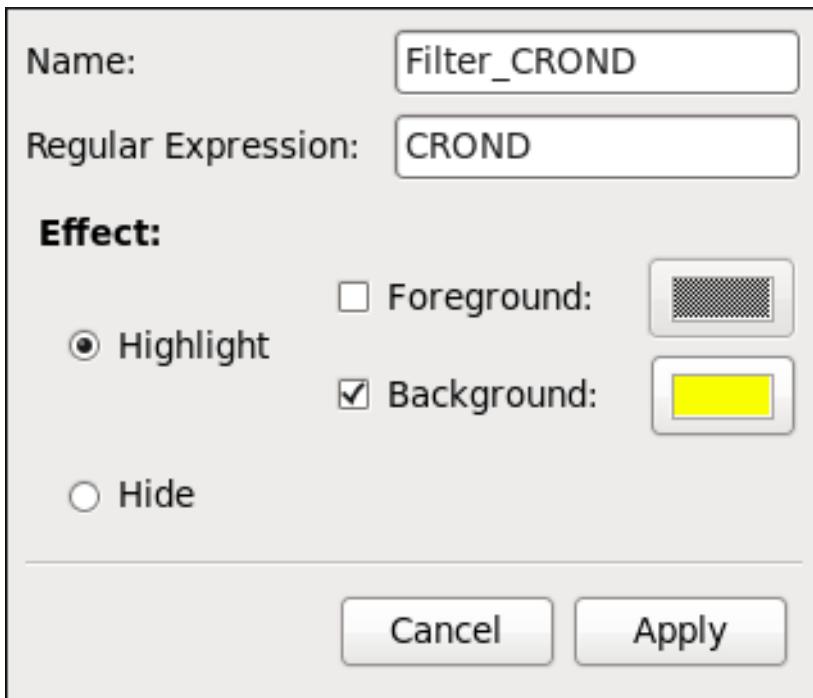


Figure 25.4. Log File Viewer - defining a filter

When defining a filter, the following parameters can be edited:

- **Name** – Specifies the name of the filter.
- **Regular Expression** – Specifies the regular expression that will be applied to the log file and will attempt to match any possible strings of text in it.
- **Effect**

- **Highlight** – If checked, the found results will be highlighted with the selected color. You may select whether to highlight the background or the foreground of the text.
- **Hide** – If checked, the found results will be hidden from the log file you are viewing.

When you have at least one filter defined, it can be selected from the **Filters** menu and it will automatically search for the strings you have defined in the filter and highlight or hide every successful match in the log file you are currently viewing.

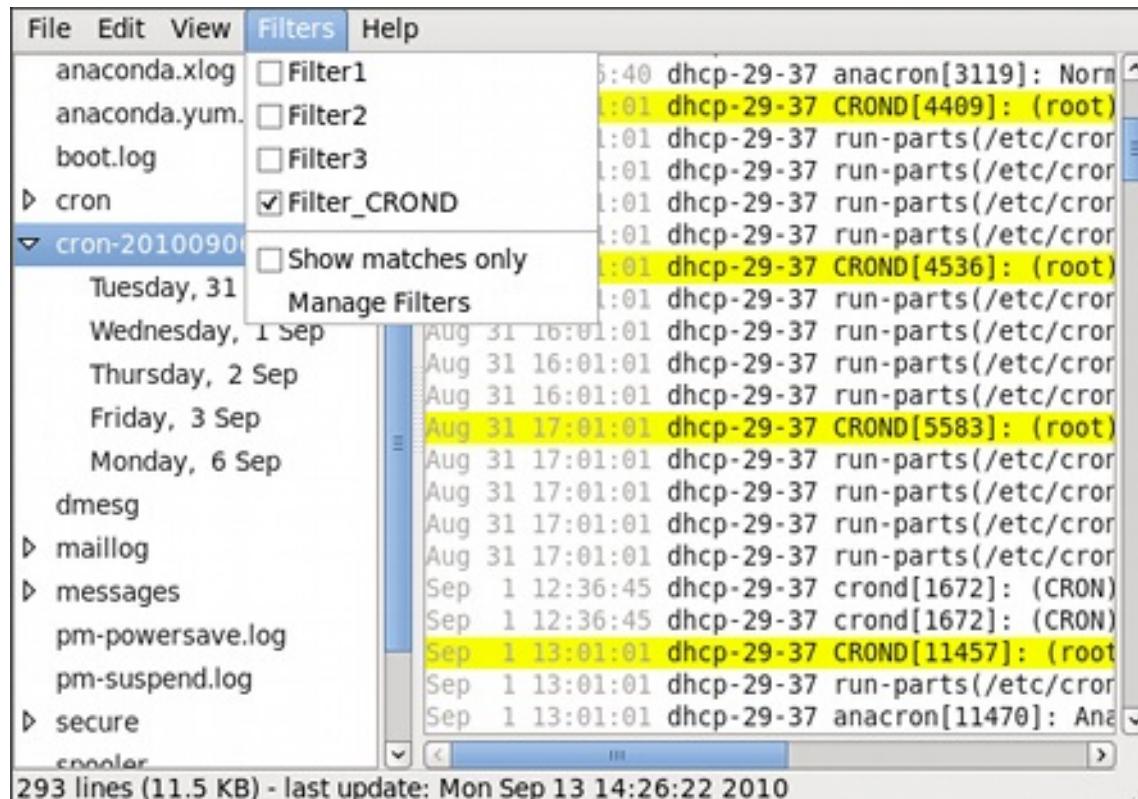


Figure 25.5. Log File Viewer - enabling a filter

When you select the **Show matches only** option, only the matched strings will be shown in the log file you are currently viewing.

25.9.2. Adding a Log File

To add a log file you want to view in the list, select **File → Open**. This will display the **Open Log** window where you can select the directory and file name of the log file you want to view. [Figure 25.6, “Log File Viewer - adding a log file”](#) illustrates the **Open Log** window.

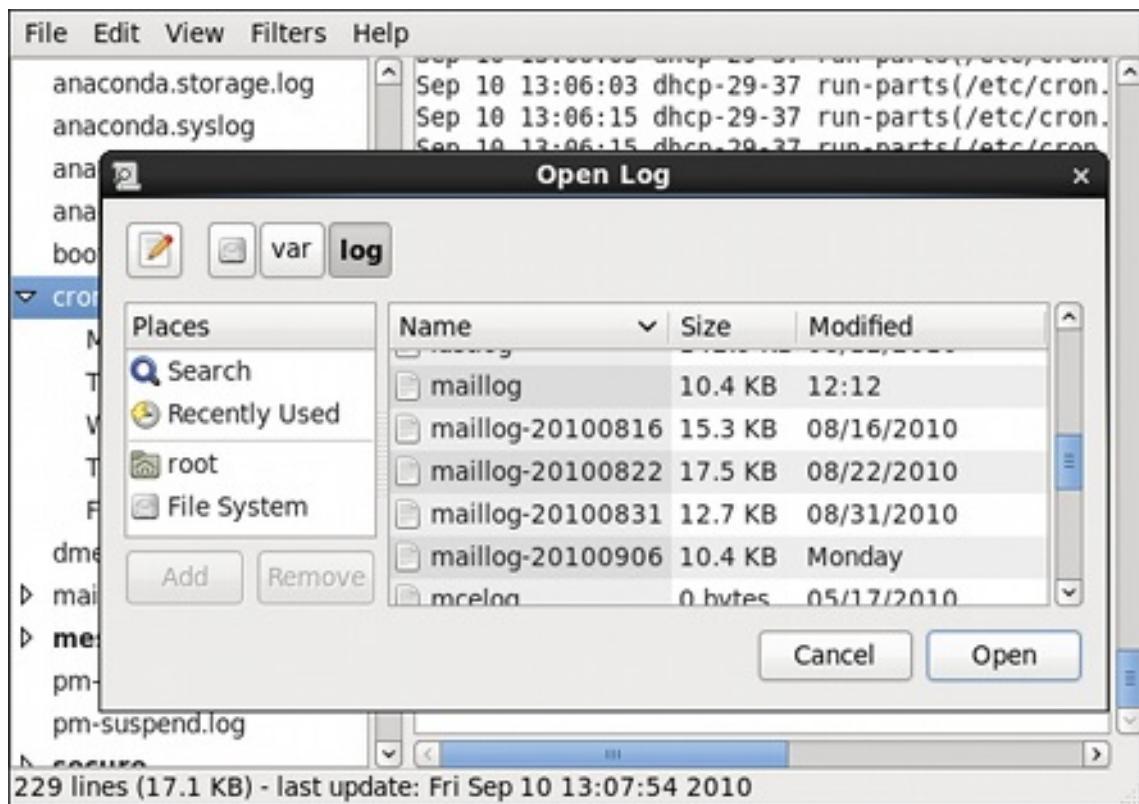


Figure 25.6. Log File Viewer – adding a log file

Click on the **Open** button to open the file. The file is immediately added to the viewing list where you can select it and view its contents.



NOTE

The **Log File Viewer** also allows you to open log files zipped in the **.gz** format.

25.9.3. Monitoring Log Files

Log File Viewer monitors all opened logs by default. If a new line is added to a monitored log file, the log name appears in bold in the log list. If the log file is selected or displayed, the new lines appear in bold at the bottom of the log file. [Figure 25.7, “Log File Viewer – new log alert”](#) illustrates a new alert in the **cron** log file and in the **messages** log file. Clicking on the **cron** log file displays the logs in the file with the new lines in bold.

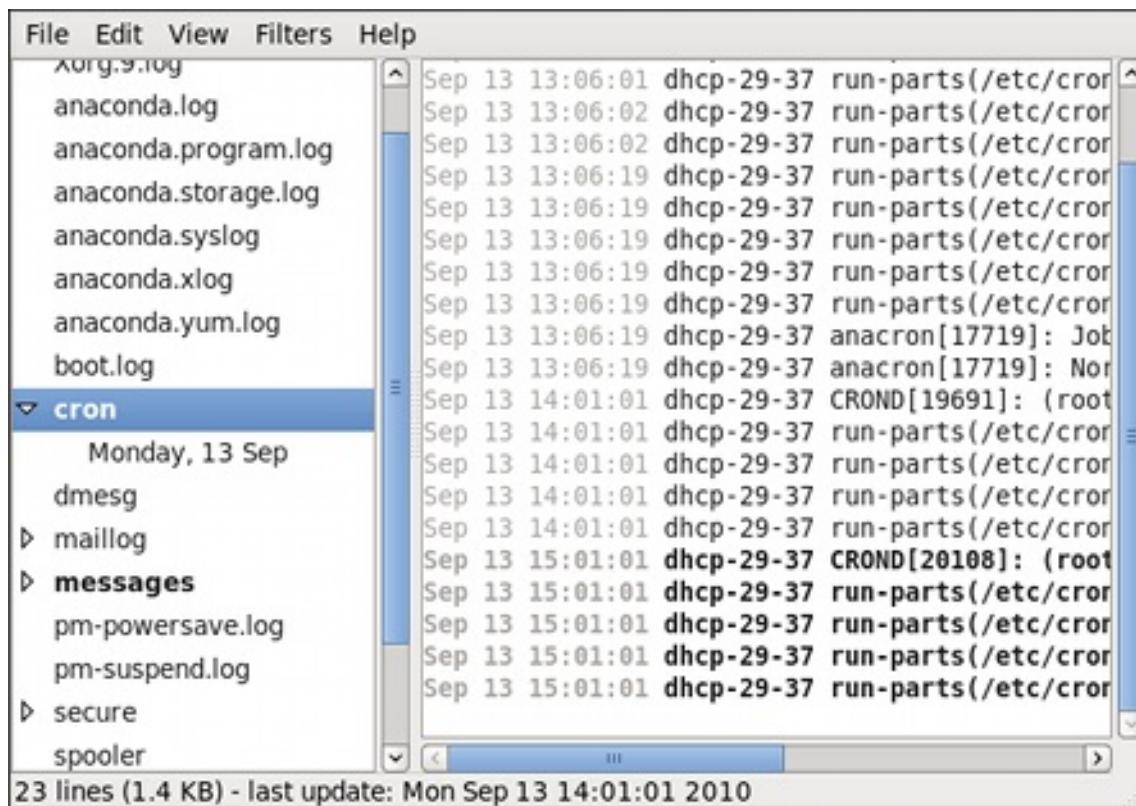


Figure 25.7. Log File Viewer - new log alert

25.10. ADDITIONAL RESOURCES

For more information on how to configure the **rsyslog** daemon and how to locate, view, and monitor log files, see the resources listed below.

Installed Documentation

- **rsyslogd(8)** – The manual page for the **rsyslogd** daemon documents its usage.
- **rsyslog.conf(5)** – The manual page named **rsyslog.conf** documents available configuration options.
- **logrotate(8)** – The manual page for the **logrotate** utility explains in greater detail how to configure and use it.

Online Documentation

The **rsyslog** home page offers additional documentation, configuration examples, and video tutorials. Make sure to consult the documents relevant to the version you are using:

- [rsyslog version 5 documentation on the rsyslog home page](#) – The default version of **rsyslog** in Red Hat Enterprise Linux 6 is version 5.
- [rsyslog version 7 documentation on the rsyslog home page](#) – Version 7 of **rsyslog** is available for Red Hat Enterprise Linux 6 in the **rsyslog7** package.
- [Description of queues on the rsyslog Home Page](#) – General information on various types of message queues and their usage.

See Also

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

and **sudo** commands.

CHAPTER 26. UPGRADING MYSQL

Red Hat is committed to fully supporting the upstream version of MySQL, which is currently included in Red Hat Enterprise Linux, until the end of Maintenance Support 2 Phase, as long as upstream security and bug fixes are available. For overview of Red Hat Enterprise Linux Life Cycle, see https://access.redhat.com/support/policy/updates/errata#Maintenance_Support_2_Phase.

More recent versions of MySQL, MySQL 5.6 and MySQL 5.7, are provided as the rh-mysql56 and rh-mysql57 Software Collections. These components are part of Red Hat Software Collections, available for all supported releases of Red Hat Enterprise Linux 6 on AMD64 and Intel 64 architectures.

For information on how to get access to Red Hat Software Collections, see the [Red Hat Software Collections Release Notes](#).

See the [Red Hat Software Collections Product Life Cycle](#) document for information regarding length of support for individual components.

Note that you cannot directly migrate from MySQL 5.1 to the currently supported versions. Refer to detailed procedures how to migrate [from MySQL 5.1 to MySQL 5.5](#), [from MySQL 5.5 to MySQL 5.6](#), and [from MySQL 5.6 to MySQL 5.7](#).

CHAPTER 27. AUTOMATING SYSTEM TASKS

Tasks, also known as *jobs*, can be configured to run automatically within a specified period of time, on a specified date, or when the system load average decreases below 0.8.

Red Hat Enterprise Linux is pre-configured to run important system tasks to keep the system updated. For example, the slocate database used by the **locate** command is updated daily. A system administrator can use automated tasks to perform periodic backups, monitor the system, run custom scripts, and so on.

Red Hat Enterprise Linux comes with the following automated task utilities: **cron**, **anacron**, **at**, and **batch**.

Every utility is intended for scheduling a different job type: while Cron and Anacron schedule recurring jobs, At and Batch schedule one-time jobs (see [Section 27.1, “Cron and Anacron”](#) and [Section 27.2, “At and Batch”](#) respectively).

27.1. CRON AND ANACRON

Both, Cron and Anacron, are daemons that can schedule execution of recurring tasks to a certain point in time defined by the exact time, day of the month, month, day of the week, and week.

Cron jobs can run as often as every minute. However, the utility assumes that the system is running continuously and if the system is not on at the time when a job is scheduled, the job is not executed.

On the other hand, Anacron remembers the scheduled jobs if the system is not running at the time when the job is scheduled. The job is then executed as soon as the system is up. However, Anacron can only run a job once a day.

27.1.1. Installing Cron and Anacron

To install Cron and Anacron, you need to install the `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 **rpm -q cronie cronie-anacron** command. 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 the packages, use the **yum** command in the following form:

```
yum install package
```

For example, to install both Cron and Anacron, type the following at a shell prompt:

```
~]# yum install cronie cronie-anacron
```

Note that you must have superuser privileges (that is, you must be logged in as **root**) to run this command. For more information on how to install new packages in Red Hat Enterprise Linux, see [Section 8.2.4, “Installing Packages”](#).

27.1.2. Running the Crond Service

The cron and anacron jobs are both picked by the **crond** service. This section provides information on how to start, stop, and restart the **crond** service, and shows how to enable it in a particular runlevel. For

more information on the concept of runlevels and how to manage system services in Red Hat Enterprise Linux in general, see [Chapter 12, Services and Daemons](#).

27.1.2.1. Starting and Stopping the Cron Service

To determine if the service is running, use the command **service crond status**.

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

```
service crond start
```

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

```
chkconfig crond on
```

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

27.1.2.2. Stopping the Cron Service

To stop the **crond** service, type the following at a shell prompt as **root**

```
service crond stop
```

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

```
chkconfig crond off
```

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

27.1.2.3. Restarting the Cron Service

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

```
service crond restart
```

This command stops the service and starts it again in quick succession.

27.1.3. Configuring Anacron Jobs

The main configuration file to schedule jobs is the **/etc/anacrontab** file, which can be only accessed by the root user. The file contains the following:

```
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=45
# the jobs will be started during the following hours only
START_HOURS_RANGE=3-22

#period in days  delay in minutes  job-identifier  command
```

```

1      5    cron.daily  nice run-parts /etc/cron.daily
7      25   cron.weekly nice run-parts /etc/cron.weekly
@monthly 45   cron.monthly nice run-parts /etc/cron.monthly

```

The first three lines define the variables that configure the environment in which the anacron tasks run:

- **SHELL** – shell environment used for running jobs (in the example, the Bash shell)
- **PATH** – paths to executable programs
- **MAILTO** – user name of the user who receives the output of the anacron jobs by email

If the **MAILTO** variable is not defined (**MAILTO=**), the email is not sent.

The next two variables modify the scheduled time for the defined jobs:

- **RANDOM_DELAY** – maximum number of minutes that will be added to the **delay in minutes** variable which is specified for each job

The minimum delay value is set, by default, to 6 minutes.

If **RANDOM_DELAY** is, for example, set to **12**, then between 6 and 12 minutes are added to the **delay in minutes** for each job in that particular anacrontab. **RANDOM_DELAY** can also be set to a value below **6**, including **0**. When set to **0**, no random delay is added. This proves to be useful when, for example, more computers that share one network connection need to download the same data every day.

- **START_HOURS_RANGE** – interval, when scheduled jobs can be run, in hours

In case the time interval is missed, for example due to a power failure, the scheduled jobs are not executed that day.

The remaining lines in the **/etc/anacrontab** file represent scheduled jobs and follow this format:

```
period in days  delay in minutes  job-identifier  command
```

- **period in days** – frequency of job execution in days

The property value can be defined as an integer or a macro (**@daily**, **@weekly**, **@monthly**), where **@daily** denotes the same value as integer 1, **@weekly** the same as 7, and **@monthly** specifies that the job is run once a month regardless of the length of the month.

- **delay in minutes** – number of minutes anacron waits before executing the job

The property value is defined as an integer. If the value is set to **0**, no delay applies.

- **job-identifier** – unique name referring to a particular job used in the log files
- **command** – command to be executed

The command can be either a command such as **ls /proc >> /tmp/proc** or a command which executes a custom script.

Any lines that begin with a hash sign (#) are comments and are not processed.

27.1.3.1. Examples of Anacron Jobs

The following example shows a simple **/etc/anacrontab** file:

```
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=30
# the jobs will be started during the following hours only
START_HOURS_RANGE=16-20

#period in days  delay in minutes  job-identifier  command
1      20  dailyjob    nice run-parts /etc/cron.daily
7      25  weeklyjob   /etc/weeklyjob.bash
@monthly 45  monthlyjob  ls /proc >> /tmp/proc
```

All jobs defined in this **anacrontab** file are randomly delayed by 6–30 minutes and can be executed between 16:00 and 20:00.

The first defined job is triggered daily between 16:26 and 16:50 (RANDOM_DELAY is between 6 and 30 minutes; the delay in minutes property adds 20 minutes). The command specified for this job executes all present programs in the **/etc/cron.daily** directory using the **run-parts** script (the **run-parts** scripts accepts a directory as a command-line argument and sequentially executes every program in the directory).

The second job executes the **weeklyjob.bash** script in the **/etc** directory once a week.

The third job runs a command, which writes the contents of **/proc** to the **/tmp/proc** file (**ls /proc >> /tmp/proc**) once a month.

27.1.4. Configuring Cron Jobs

The configuration file for cron jobs is the **/etc/crontab**, which can be only modified by the root user. The file contains the following:

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/
# For details see man 4 crontabs
# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .---- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .--- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | |
# * * * * * username  command to be executed
```

The first three lines contain the same variable definitions as an **anacrontab** file: **SHELL**, **PATH**, and **MAILTO**. For more information about these variables, see [Section 27.1.3, “Configuring Anacron Jobs”](#).

In addition, the file can define the **HOME** variable. The **HOME** variable defines the directory, which will be used as the home directory when executing commands or scripts run by the job.

The remaining lines in the **/etc/crontab** file represent scheduled jobs and have the following format:

```
minute hour day month day of week username command
```

The following define the time when the job is to be run:

- **minute** – any integer from 0 to 59
- **hour** – any integer from 0 to 23
- **day** – any integer from 1 to 31 (must be a valid day if a month is specified)
- **month** – any integer from 1 to 12 (or the short name of the month such as jan or feb)
- **day of week** – any integer from 0 to 7, where 0 or 7 represents Sunday (or the short name of the week such as sun or mon)

The following define other job properties:

- **username** – specifies the user under which the jobs are run
- **command** – the command to be executed

The command can be either a command such as **ls /proc /tmp/proc** or a command which executes a custom script.

For any of the above values, an asterisk (*) can be used to specify all valid values. If you, for example, define the month value as an asterisk, the job will be executed every month within the constraints of the other values.

A hyphen (-) between integers specifies a range of integers. For example, **1-4** means the integers 1, 2, 3, and 4.

A list of values separated by commas (,) specifies a list. For example, **3, 4, 6, 8** indicates exactly these four integers.

The forward slash (/) can be used to specify step values. The value of an integer will be skipped within a range following the range with **/integer**. For example, minute value defined as **0-59/2** denotes every other minute in the minute field. Step values can also be used with an asterisk. For instance, if the month value is defined as ***/3**, the task will run every third month.

Any lines that begin with a hash sign (#) are comments and are not processed.

Users other than root can configure cron tasks with the **crontab** utility. The user-defined crontabs are stored in the **/var/spool/cron/** directory and executed as if run by the users that created them.

To create a crontab as a user, login as that user and type the command **crontab -e** to edit the user's crontab with the editor specified in the **VISUAL** or **EDITOR** environment variable. The file uses the same format as **/etc/crontab**. When the changes to the crontab are saved, the crontab is stored according to user name and written to the file **/var/spool/cron/username**. To list the contents of your crontab file, use the **crontab -l** command.



NOTE

Do not specify the user when defining a job with the **crontab** utility.

The **/etc/cron.d/** directory contains files that have the same syntax as the **/etc/crontab** file. Only root is allowed to create and modify files in this directory.



NOTE

The cron daemon checks the **/etc/anacrontab** file, the **/etc/crontab** file, the **/etc/cron.d/** directory, and the **/var/spool/cron** directory every minute for changes and the detected changes are loaded into memory. It is therefore not necessary to restart the daemon after an anacrontab or a crontab file have been changed.

27.1.5. Controlling Access to Cron

To restrict the access to Cron, you can use the **/etc/cron.allow** and **/etc/cron.deny** files. These access control files use the same format with one user name on each line. Mind that no whitespace characters are permitted in either file.

If the **cron.allow** file exists, only users listed in the file are allowed to use cron, and the **cron.deny** file is ignored.

If the **cron.allow** file does not exist, users listed in the **cron.deny** file are not allowed to use Cron.

The Cron daemon (**crond**) does not have to be restarted if the access control files are modified. The access control files are checked each time a user tries to add or delete a cron job.

The root user can always use cron, regardless of the user names listed in the access control files.

You can control the access also through Pluggable Authentication Modules (PAM). The settings are stored in the **/etc/security/access.conf** file. For example, after adding the following line to the file, no other user but the root user can create crontabs:

```
-:ALL EXCEPT root :cron
```

The forbidden jobs are logged in an appropriate log file or, when using “`crontab -e`”, returned to the standard output. For more information, see **access.conf.5** (that is, **man 5 access.conf**).

27.1.6. Black and White Listing of Cron Jobs

Black and white listing of jobs is used to define parts of a job that do not need to be executed. This is useful when calling the **run-parts** script on a Cron directory, such as **/etc/cron.daily**: if the user adds programs located in the directory to the job black list, the **run-parts** script will not execute these programs.

To define a black list, create a **jobs.deny** file in the directory that **run-parts** scripts will be executing from. For example, if you need to omit a particular program from **/etc/cron.daily**, create the **/etc/cron.daily/jobs.deny** file. In this file, specify the names of the programs to be omitted from execution (only programs located in the same directory can be enlisted). If a job runs a command which runs the programs from the cron.daily directory, such as **run-parts /etc/cron.daily**, the programs defined in the **jobs.deny** file will not be executed.

To define a white list, create a **jobs.allow** file.

The principles of **jobs.deny** and **jobs.allow** are the same as those of **cron.deny** and **cron.allow** described in section [Section 27.1.5, “Controlling Access to Cron”](#).

27.2. AT AND BATCH

While Cron is used to schedule recurring tasks, the **At** utility is used to schedule a one-time task at a specific time and the **Batch** utility is used to schedule a one-time task to be executed when the system load average drops below 0.8.

27.2.1. Installing At and Batch

To determine if the at package is already installed on your system, issue the **rpm -q at** command. The command returns the full name of the at package if already installed or notifies you that the package is not available.

To install the packages, use the **yum** command in the following form:

```
yum install package
```

To install At and Batch, type the following at a shell prompt:

```
~]# yum install at
```

Note that you must have superuser privileges (that is, you must be logged in as **root**) to run this command. For more information on how to install new packages in Red Hat Enterprise Linux, see [Section 8.2.4, “Installing Packages”](#).

27.2.2. Running the At Service

The At and Batch jobs are both picked by the **atd** service. This section provides information on how to start, stop, and restart the **atd** service, and shows how to enable it in a particular runlevel. For more information on the concept of runlevels and how to manage system services in Red Hat Enterprise Linux in general, see [Chapter 12, Services and Daemons](#).

27.2.2.1. Starting and Stopping the At Service

To determine if the service is running, use the command **service atd status**.

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

```
service atd start
```

To configure the service to start automatically at boot, use the following command:

```
chkconfig atd on
```



NOTE

It is recommended to start the service at boot automatically.

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

27.2.2.2. Stopping the At Service

To stop the **atd** service, type the following at a shell prompt as **root**

```
service atd stop
```

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

```
chkconfig atd off
```

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

27.2.2.3. Restarting the At Service

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

```
service atd restart
```

This command stops the service and starts it again in quick succession.

27.2.3. Configuring an At Job

To schedule a one-time job for a specific time with the **At** utility, do the following:

1. On the command line, type the command **at *TIME***, where ***TIME*** is the time when the command is to be executed.

The ***TIME*** argument can be defined in any of the following formats:

- **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 27.2.5, "Viewing Pending Jobs"](#) for more information.

You can also restrict the usage of the **at** command. For more information, see [Section 27.2.7, "Controlling Access to At and Batch"](#) for details.

27.2.4. Configuring a Batch Job

The **Batch** application executes the defined one-time tasks when the system load average decreases below 0.8.

To define a Batch job, do the following:

1. On the command line, type the command **batch**.
2. In the displayed **at>** prompt, define the job commands:
 - Type the command the job should execute and press **Enter**. Optionally, repeat the step to provide multiple commands.
 - Enter a shell script at the prompt and press **Enter** after each line in the script.
3. Once finished, press **Ctrl+D** on an empty line to exit the prompt.

If the set of commands or the script tries to display information to standard output, the output is emailed to the user.

To view the list of pending jobs, use the **atq** command. See [Section 27.2.5, "Viewing Pending Jobs"](#) for more information.

You can also restrict the usage of the **batch** command. For more information, see [Section 27.2.7, "Controlling Access to At and Batch"](#) for details.

27.2.5. Viewing Pending Jobs

To view the pending **At** and **Batch** jobs, run the **atq** command. The **atq** command displays a list of pending jobs, with each job on a separate line. Each line follows the job number, date, hour, job class, and user name format. Users can only view their own jobs. If the root user executes the **atq** command, all jobs for all users are displayed.

27.2.6. Additional Command-Line Options

Additional command-line options for **at** and **batch** include the following:

Table 27.1. at and batch Command-Line Options

Option	Description
-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.

27.2.7. Controlling Access to At and Batch

You can restrict the access to the **at** and **batch** commands using the **/etc/at.allow** and **/etc/at.deny** files. These access control files use the same format defining one user name on each line. Mind that no whitespace are permitted in either file.

If the file **at.allow** exists, only users listed in the file are allowed to use **at** or **batch**, and the **at.deny** file is ignored.

If **at.allow** does not exist, users listed in **at.deny** are not allowed to use **at** or **batch**.

The **at** daemon (**atd**) does not have to be restarted if the access control files are modified. The access control files are read each time a user tries to execute the **at** or **batch** commands.

The root user can always execute **at** and **batch** commands, regardless of the content of the access control files.

27.3. ADDITIONAL RESOURCES

To learn more about configuring automated tasks, see the following installed documentation:

- **cron** man page contains an overview of cron.
- **crontab** man pages in sections 1 and 5:
 - The manual page in section 1 contains an overview of the **crontab** file.
 - The man page in section 5 contains the format for the file and some example entries.
- **anacron** manual page contains an overview of anacron.
- **anacrontab** manual page contains an overview of the **anacrontab** file.
- **/usr/share/doc/at-<version>/timespec** contains detailed information about the time values that can be used in cron job definitions.
- **at** manual page contains descriptions of **at** and **batch** and their command-line options.

CHAPTER 28. AUTOMATIC BUG REPORTING TOOL (ABRT)

The **Automatic Bug Reporting Tool**, commonly abbreviated as **ABRT**, consists of the **abrtd** daemon and a number of system services and utilities to process, analyze, and report detected problems. The daemon runs silently in the background most of the time, and springs into action when an application crashes or a kernel oops is detected. The daemon then collects the relevant problem data such as a core file if there is one, the crashing application's command-line parameters, and other data of forensic utility. For a brief overview of the most important ABRT components, see [Table 28.1, "Basic ABRT components"](#).



IMPORTANT

For Red Hat Enterprise Linux 6.2, the Automatic Bug Reporting Tool has been upgraded to version 2.0. The **ABRT** 2-series brings major improvements to automatic bug detection and reporting.

Table 28.1. Basic ABRT components

Component	Package	Description
abrt	abrt	The ABRT daemon which runs under the root user as a background service.
abrt-applet	abrt-gui	The program that receives messages from abrtd and informs you whenever a new problem occurs.
abrt-gui	abrt-gui	The GUI application that shows collected problem data and allows you to further process it.
abrt-cli	abrt-cli	The command-line interface that provides similar functionality to the GUI.
abrt-ccpp	abrt-addon-ccpp	The ABRT service that provides the C/C++ problems analyzer.
abrt-oops	abrt-addon-kerneloops	The ABRT service that provides the kernel oopses analyzer.
abrt-vmcore	abrt-addon-vmcore ^[a]	The ABRT service that provides the kernel panic analyzer and reporter.

[a] The abrt-addon-vmcore package is provided by the Optional subscription channel. See [Section 8.4.8, "Adding the Optional and Supplementary Repositories"](#) for more information on Red Hat additional channels.

ABRT currently supports detection of crashes in applications written in the C/C++ and Python languages, as well as kernel oopses. With Red Hat Enterprise Linux 6.3, **ABRT** can also detect kernel panics if the additional abrt-addon-vmcore package is installed and the **kdump** crash dumping mechanism is enabled and configured on the system accordingly.

ABRT is capable of reporting problems to a remote issue tracker. Reporting can be configured to

happen automatically whenever an issue is detected, or problem data can be stored locally, reviewed, reported, and deleted manually by a user. The reporting tools can send problem data to a Bugzilla database, a Red Hat Technical Support (RHTSupport) site, upload it using **FTP/SCP**, email it, or write it to a file.

The part of **ABRT** which handles already-existing problem data (as opposed to, for example, creation of new problem data) has been factored out into a separate project, **libreport**. The **libreport** library provides a generic mechanism for analyzing and reporting problems, and it is used by applications other than **ABRT**. However, **ABRT** and **libreport** operation and configuration is closely integrated. They are therefore discussed as one in this document.

Whenever a problem is detected, **ABRT** compares it with all existing problem data and determines whether that same problem has been recorded. If it has been, the existing problem data is updated and the most recent (duplicate) problem is not recorded again. If this problem is not recognized by **ABRT**, a **problem data directory** is created. A problem data directory typically consists of files such as: **analyzer**, **architecture**, **coredump**, **cmdline**, **executable**, **kernel**, **os_release**, **reason**, **time** and **uid**.

Other files, such as **backtrace**, can be created during analysis depending on which analyzer method is used and its configuration settings. Each of these files holds specific information about the system and the problem itself. For example, the **kernel** file records the version of the crashed kernel.

After the problem directory is created and problem data gathered, you can further process, analyze and report the problem using either the **ABRT** GUI, or the **abrt-cli** utility for the command line. For more information about these tools, see [Section 28.2, “Using the Graphical User Interface”](#) and [Section 28.3, “Using the Command-Line Interface”](#) respectively.



NOTE

If you do not use **ABRT** to further analyze and report the detected problems but instead you report problems using a legacy problem reporting tool, **report**, note that you can no longer file new bugs. The **report** utility can now only be used to attach new content to the already existing bugs in the RHTSupport or Bugzilla database. Use the following command to do so:

report [-v] --target target --ticket ID file

...where *target* is either **strata** for reporting to RHTSupport or **bugzilla** for reporting to Bugzilla. *ID* stands for number identifying an existing problem case in the respective database, and *file* is a file containing information to be added to the problem case.

If you want to report new problems and you do not want to use **abrt-cli**, you can now use the **report-cli** utility instead of **report**. Issue the following command to let **report-cli** to guide you through the problem reporting process:

report-cli -r dump_directory

...where *dump_directory* is a problem data directory created by **ABRT** or some other application using **libreport**. For more information on **report-cli**, see **man report-cli**.

28.1. INSTALLING ABRT AND STARTING ITS SERVICES

As a prerequisite for its use, the **abrt** daemon requires the **abrt** user to exist for file system operations in the **/var/spool/abrt** directory. When the **abrt** package is installed, it automatically creates the **abrt** user whose UID and GID is 173, if such user does not already exist. Otherwise, the **abrt** user can be

created manually. In that case, any UID and GID can be chosen, because **abrtd** does not require a specific UID and GID.

As the first step in order to use **ABRT**, you should ensure that the abrt-desktop package is installed on your system by running the following command as the root user:

```
~]# yum install abrt-desktop
```

With abrt-desktop installed, you will be able to use **ABRT** only in its graphical interface. If you intend to use **ABRT** on the command line, install the abrt-cli package:

```
~]# yum install abrt-cli
```

See [Section 8.2.4, “Installing Packages”](#) for more information on how to install packages with the **Yum** package manager.

Your next step should be to verify that **abrtd** is running. The daemon is typically configured to start up at boot time. You can use the following command as root to verify its current status:

```
~]# service abrt status  
abrtd (pid 1535) is running...
```

If the **service** command returns the **abrt is stopped** message, the daemon is not running. It can be started for the current session by entering this command:

```
~]# service abrt start  
Starting abrt daemon: [ OK ]
```

Similarly, you can follow the same steps to check and start up the **abrt-ccpp** service if you want **ABRT** to catch C/C++ crashes. To set **ABRT** to detect kernel oopses, use the same steps for the **abrt-oops** service. Note that this service cannot catch kernel oopses which cause the system to fail, to become unresponsive or to reboot immediately. To be able to detect such kernel oopses with **ABRT**, you need to install the **abrt-vmcore** service. If you require this functionality, see [Section 28.4.5, “Configuring ABRT to Detect a Kernel Panic”](#) for more information.

When installing **ABRT** packages, all respective ABRT services are automatically enabled for **runlevels 3 and 5**. You can disable or enable any ABRT service for the desired runlevels using the **chkconfig** utility. See [Section 12.2.3, “Using the chkconfig Utility”](#) for more information.



WARNING

Please note that installing **ABRT** packages overwrites the **/proc/sys/kernel/core_pattern** file which can contain a template used to name core dump files. The content of this file will be overwritten to:

```
~]# /usr/libexec/abrt-hook-ccpp %s %c %p %u %g %t e
```

Finally, if you run ABRT in a graphical desktop environment, you can verify that the **ABRT notification applet** is running:

```
~]$ ps -el | grep abrt-applet
0 S 500 2036 1824 0 80 0 - 61604 poll_s ? 00:00:00 abrt-applet
```

If the **ABRT** notification applet is not running, you can start it manually in your current desktop session by running the **abrt-applet** program:

```
~]$ abrt-applet &
[1] 2261
```

The applet can be configured to start automatically when your graphical desktop session starts. You can ensure that the **ABRT** notification applet is added to the list of programs and selected to run at system startup by selecting the **System → Preferences → Startup Applications** menu in the top panel.

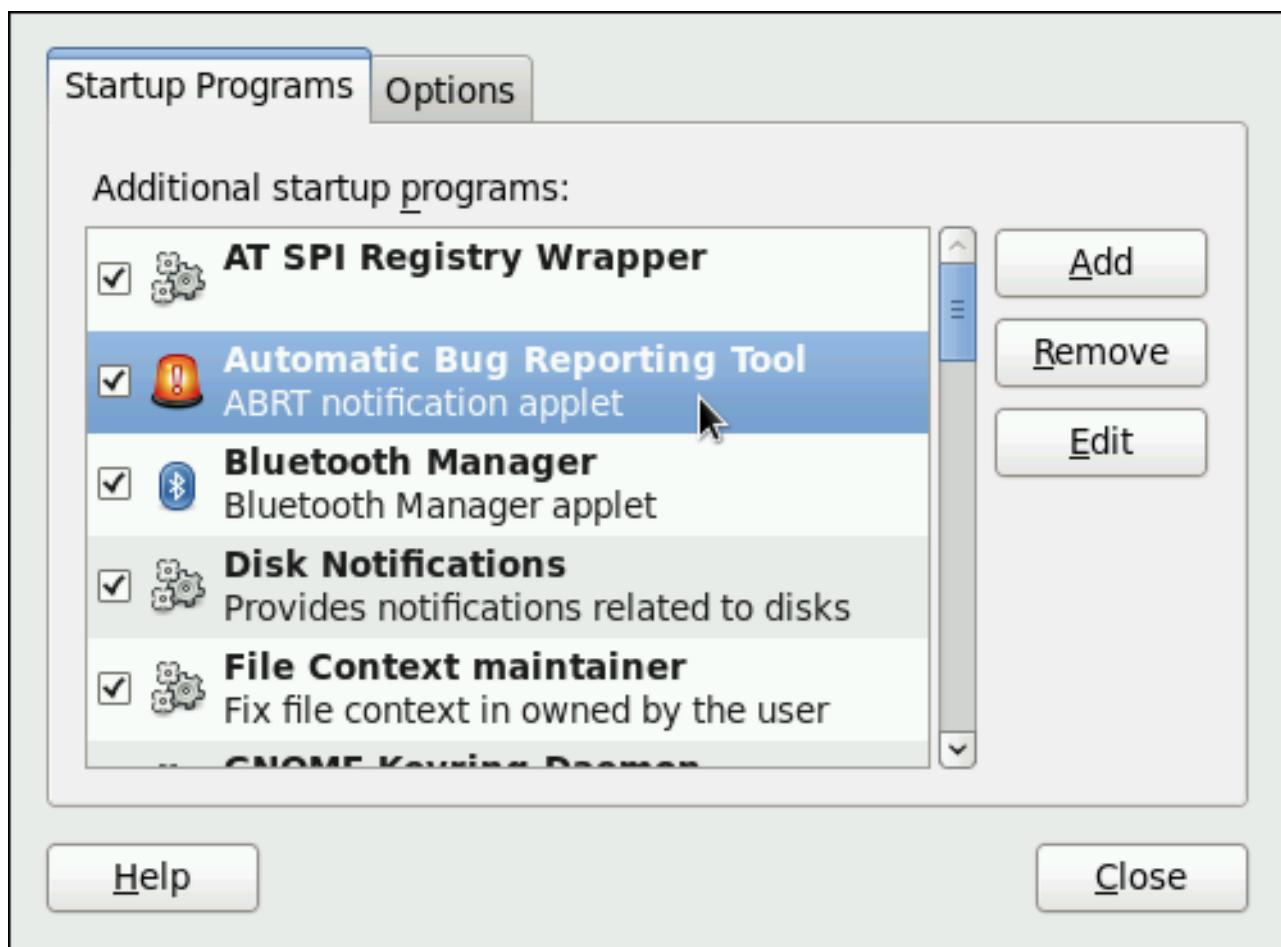


Figure 28.1. Setting ABRT notification applet to run automatically.

28.2. USING THE GRAPHICAL USER INTERFACE

The **ABRT** daemon sends a broadcast D-Bus message whenever a problem report is created. If the **ABRT** notification applet is running, it catches this message and displays an orange alarm icon in the Notification Area. You can open the **ABRT GUI** application using this icon. As an alternative, you can display the **ABRT GUI** by selecting the **Application → System Tools → Automatic Bug Reporting Tool** menu item.

Alternatively, you can run the **ABRT GUI** from the command line as follows:

```
~]$ abrt-gui &
```

The **ABRT** GUI provides an easy and intuitive way of viewing, reporting and deleting of reported problems. The **ABRT** window displays a list of detected problems. Each problem entry consists of the name of the failing application, the reason why the application crashed, and the date of the last occurrence of the problem.

The screenshot shows a window titled 'Report' with a menu bar containing 'Report', 'Edit', and 'Help'. Below the menu is a table with three columns: 'Source', 'Problem', and 'Last Occurrence'. The 'Source' column lists the failing application names, the 'Problem' column lists the crash reason, and the 'Last Occurrence' column lists the date and time of the most recent crash. The table has a header row and several data rows. At the bottom of the window are two buttons: 'Delete' and 'Open'.

Source	Problem	Last Occurrence
gnote	Process /usr/bin/gnote was killed by signal 11 (SIGSEGV)	2011-09-18 18:02
gedit	Process /usr/bin/gedit was killed by signal 11 (SIGSEGV)	2011-09-18 15:59
gcalctool	Process /usr/bin/gcalctool was killed by signal 11 (SIGSEGV)	2011-09-18 15:56
gdb	Process /usr/bin/gdb was killed by signal 11 (SIGSEGV)	2011-09-13 10:18
firefox	Process /usr/lib64/firefox-3.6/firefox was killed by signal 11 (SIGSEGV)	2011-09-13 09:59
coreutils	Process /bin/sleep was killed by signal 11 (SIGSEGV)	2011-09-12 18:37

Figure 28.2. An example of running ABRT GUI.

If you double-click on a problem report line, you can access the detailed problem description and proceed with the process of determining how the problem should be analyzed, and where it should be reported.

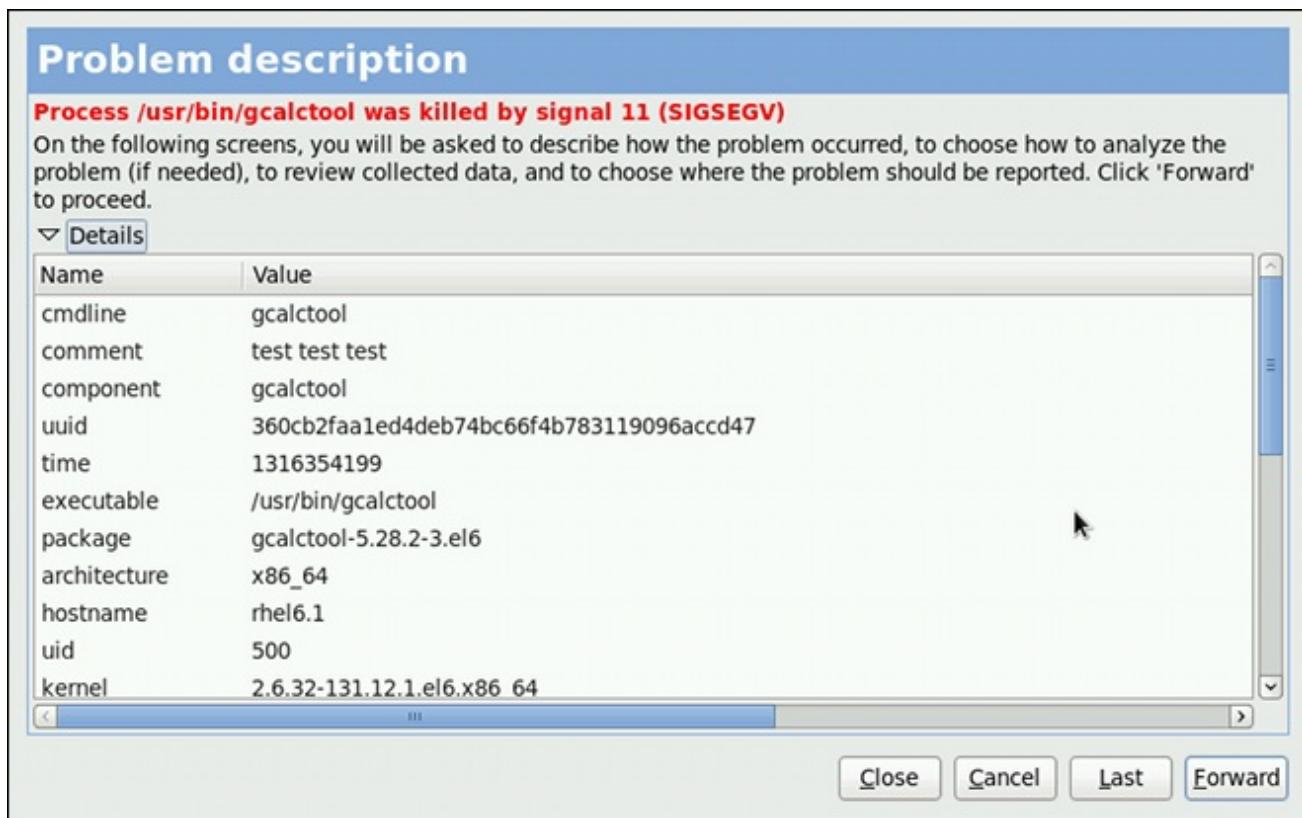


Figure 28.3. A detailed problem data example

You are first asked to provide additional information about the problem which occurred. You should provide detailed information on how the problem happened and what steps should be done in order to reproduce it. In the next steps, choose how the problem will be analyzed and generate a backtrace depending on your configuration. You can skip the analysis and backtrace-generation steps but remember that developers need as much information about the problem as possible. You can always modify the backtrace and remove any sensitive information you do not want to provide before you send the problem data out.

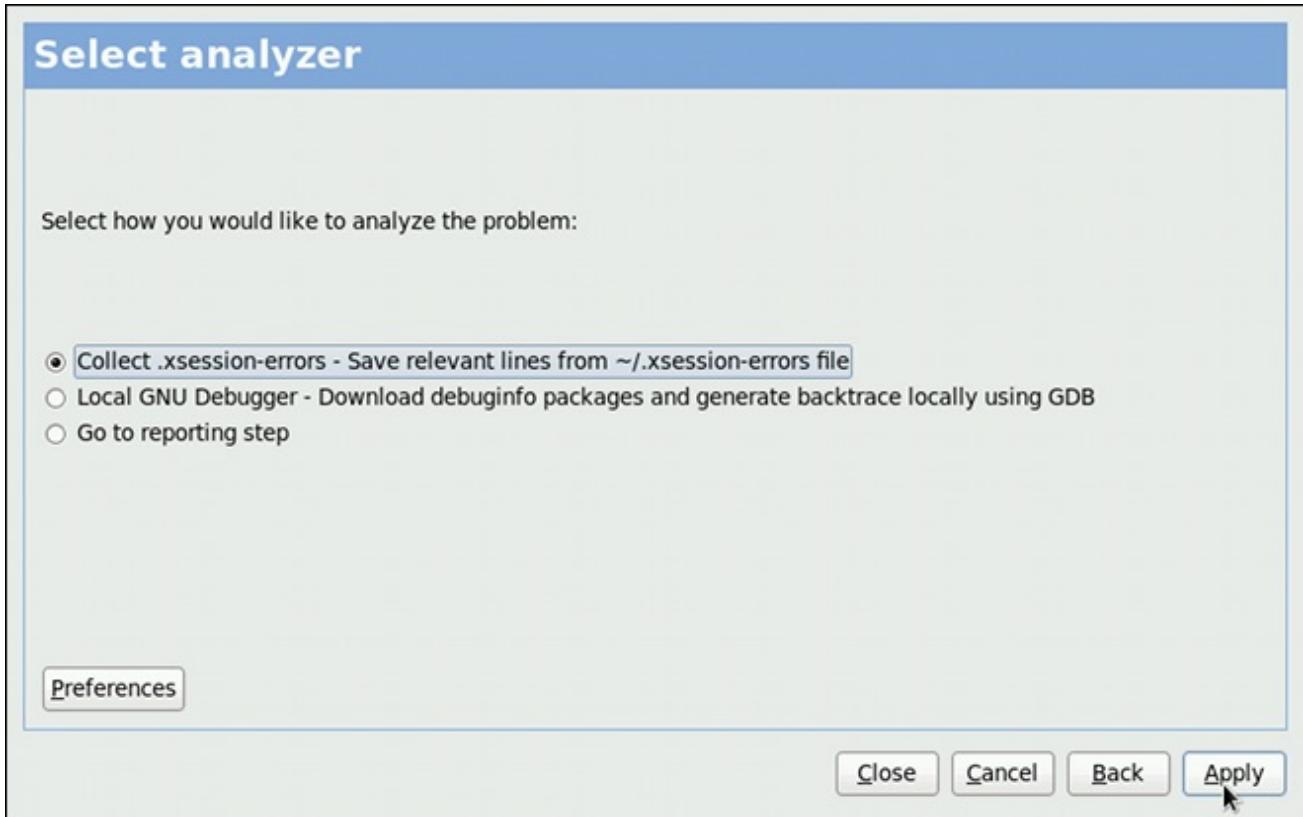


Figure 28.4. Selecting how to analyze the problem

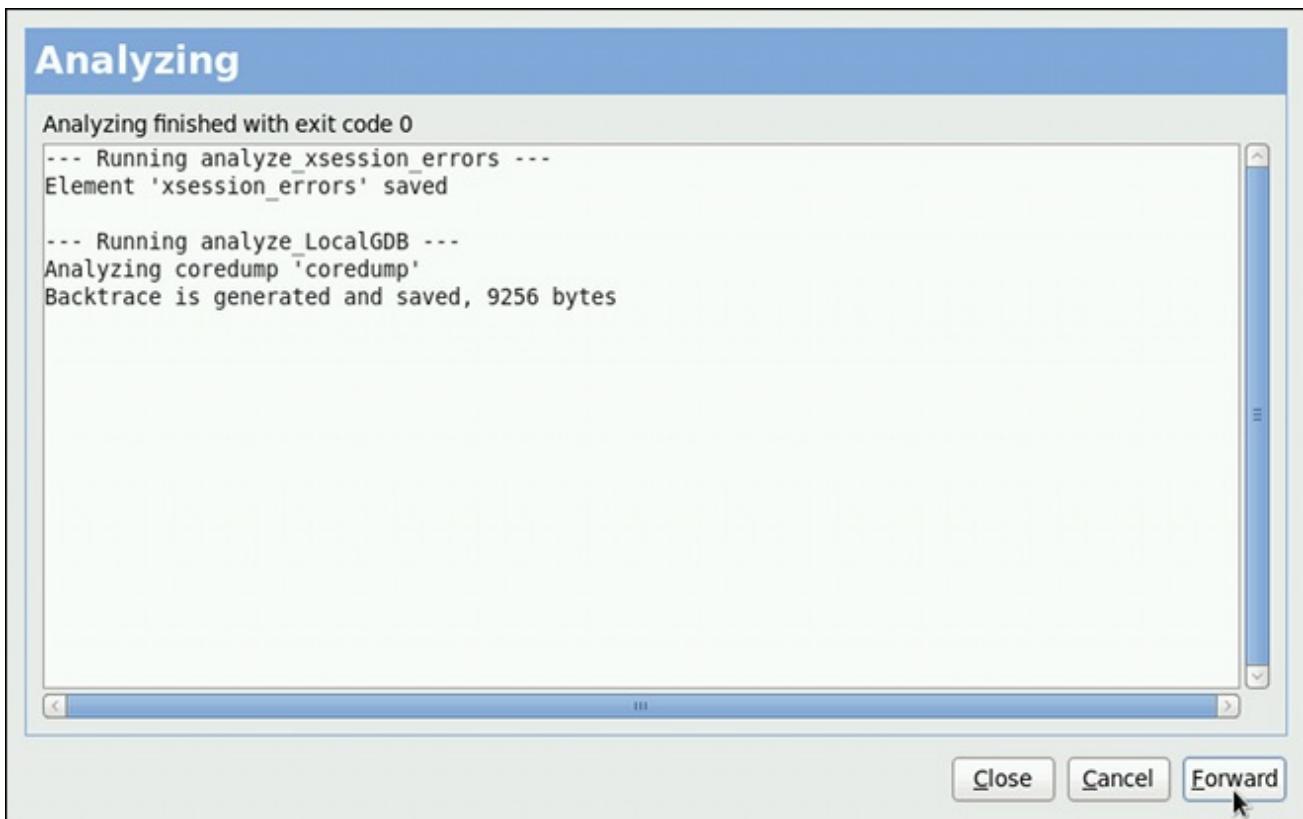


Figure 28.5. ABRT analyzing the problem

Next, choose how you want to report the issue. If you are using Red Hat Enterprise Linux, Red Hat Customer Support is the preferred choice.



Figure 28.6. Selecting a problem reporter

If you choose to report to Red Hat Customer Support, and you have not configured this event yet, you will be warned that this event is not configured properly and you will be offered an option to do so.



Figure 28.7. Warning - missing Red Hat Customer Support configuration

Here, you need to provide your Red Hat login information (See [Section 28.4.3, "Event Configuration in ABRT GUI"](#) for more information on how to acquire it and how to set this event.), otherwise you will fail to report the problem.

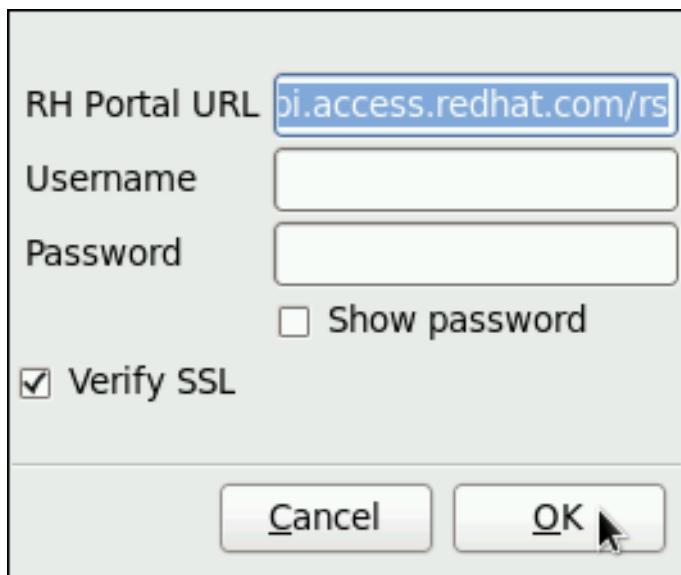


Figure 28.8. Red Hat Customer Support configuration window

After you have chosen a reporting method and have it set up correctly, review the backtrace and confirm the data to be reported.

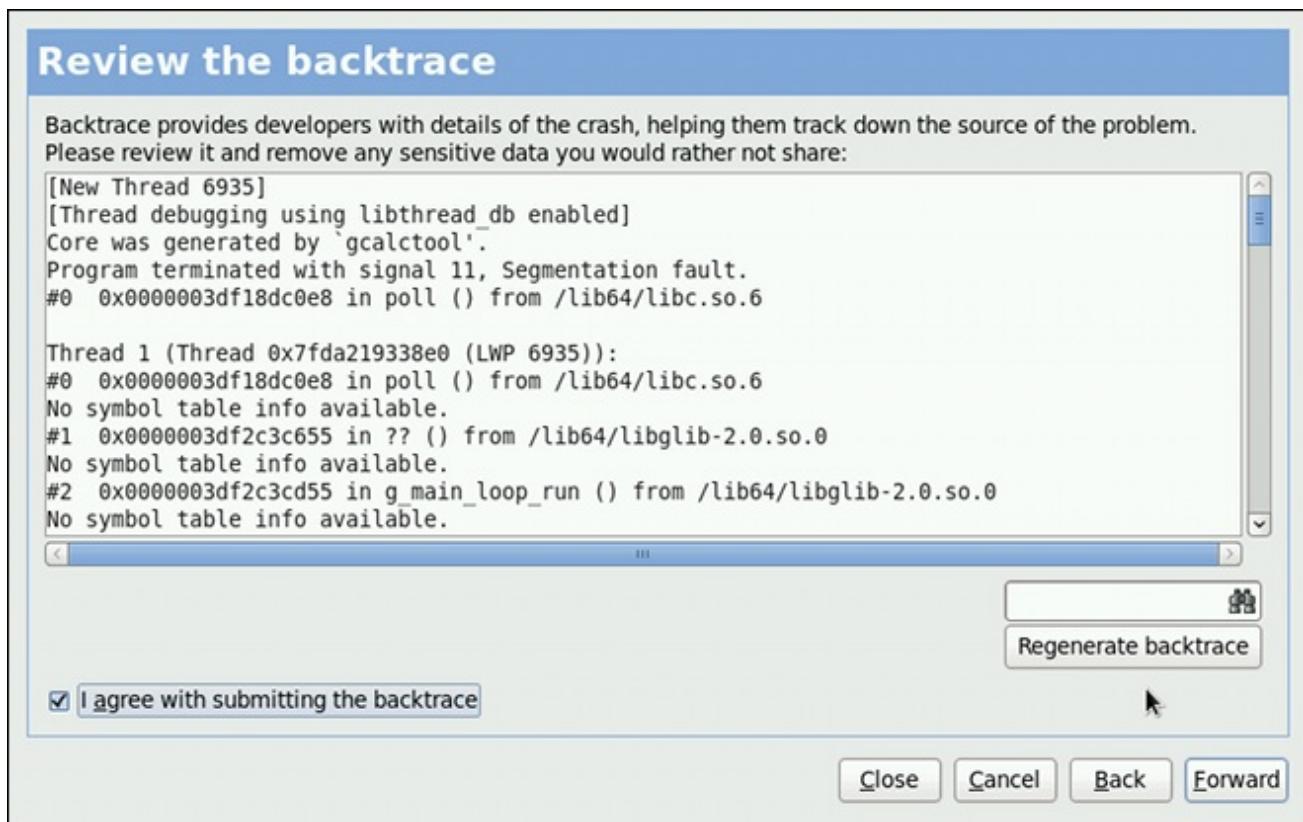


Figure 28.9. Reviewing the problem backtrace

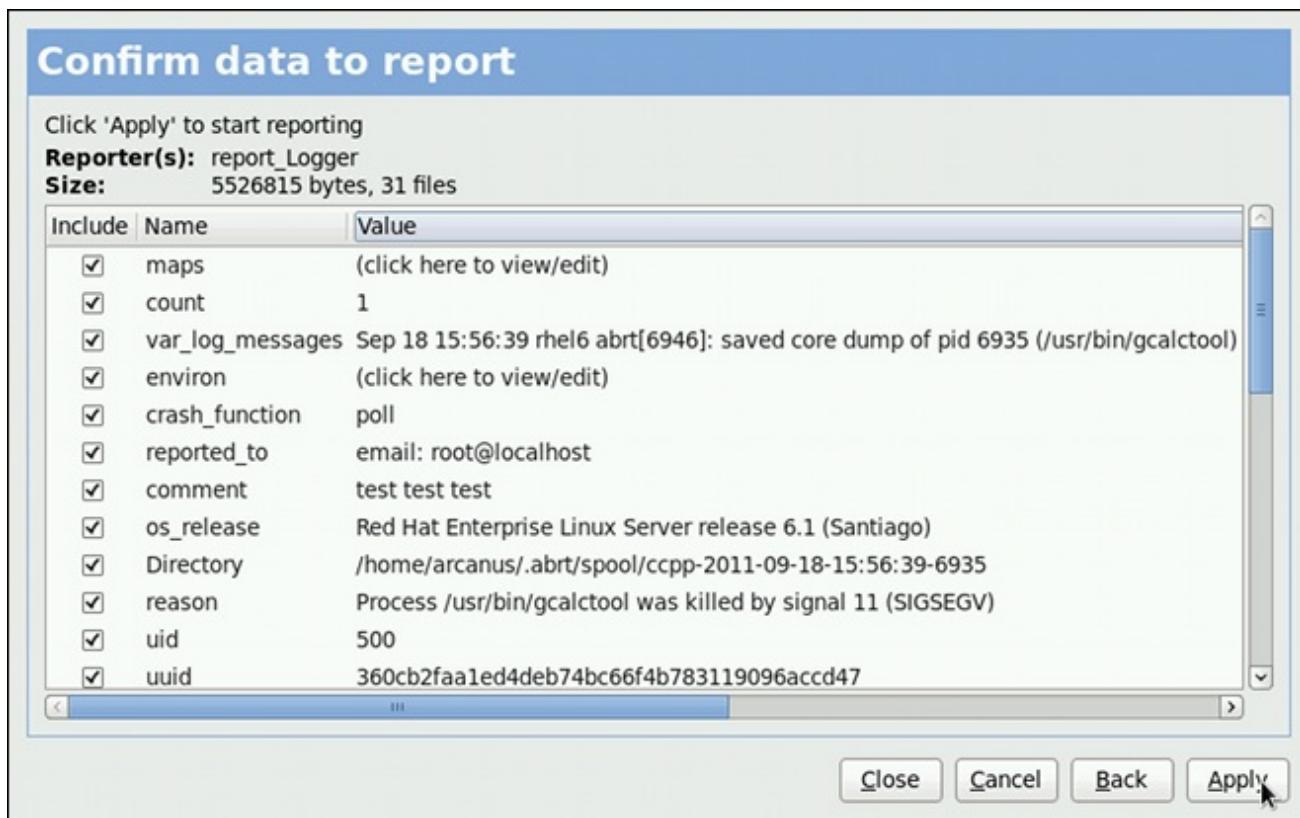


Figure 28.10. Confirming the data to report

Finally, the problem data is sent to the chosen destination, and you can now decide whether to continue with reporting the problem using another available method or finish your work on this problem. If you have reported your problem to the Red Hat Customer Support database, a problem case is filed in the database. From now on, you will be informed about the problem resolution progress via email you provided during the process of reporting. You can also oversee the problem case using the URL that is provided to you by ABRT GUI when the problem case is created, or via emails received from Red Hat Support.

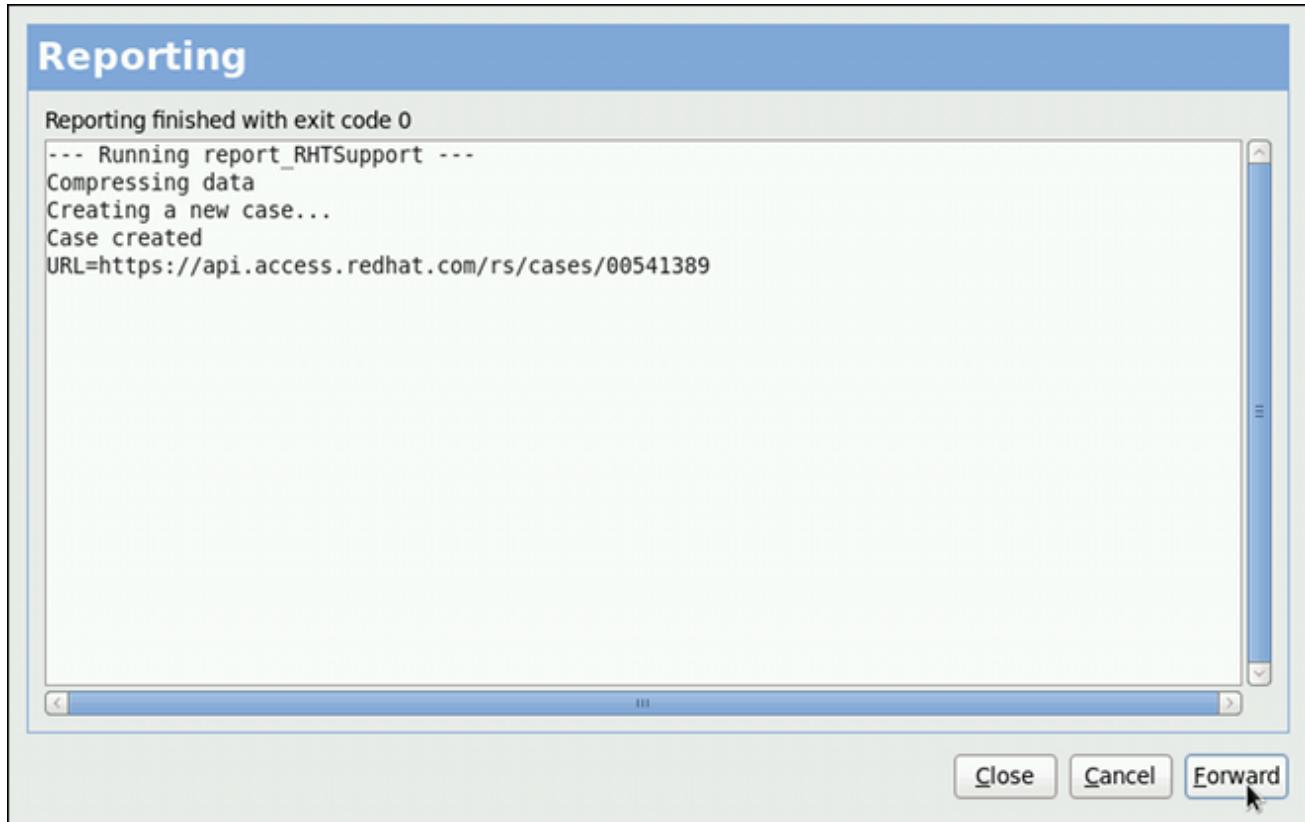


Figure 28.11. Problem is being reported to the Red Hat Customer Support database

28.3. USING THE COMMAND-LINE INTERFACE

Problem data saved by **abrtd** can be viewed, reported, and deleted using the command-line interface.

General usage of the **abrt-cli** tool can be described using the following syntax:

abrt-cli [--version] command [args]

...where *args* stands for a problem data directory and/or options modifying the commands, and *command* is one of the following sub-commands:

- **list** – lists problems and views the problem data.
- **report** – analyzes and reports problems.
- **rm** – removes unneeded problems.
- **info** – provides information about a particular problem.

To display help on particular **abrt-cli** command use:

abrt-cli command --help

The rest of the commands used with **abrt-cli** are described in the following sections.

28.3.1. Viewing Problems

To view detected problems, enter the **abrt-cli list** command:

```
~]# abrt-cli list
Directory: /var/spool/abrt/ccpp-2011-09-13-10:18:14-2895
count: 2
executable: /usr/bin/gdb
package: gdb-7.2-48.el6
time: Tue 13 Sep 2011 10:18:14 AM CEST
uid: 500

Directory: /var/spool/abrt/ccpp-2011-09-21-18:18:07-2841
count: 1
executable: /bin/bash
package: bash-4.1.2-8.el6
time: Wed 21 Sep 2011 06:18:07 PM CEST
uid: 500
```

- **Directory** – Shows the problem data directory that contains all information about the problem.
- **count** – Shows how many times this particular problem occurred.
- **executable** – Indicates which binary or executable script crashed.
- **package** – Shows the name of the package that contains the program that caused the problem.
- **time** – Shows the date and time of the last occurrence of the problem.
- **uid** – Shows the ID of the user which ran the program that crashed.

The following table shows options available with the **abrt-cli list** command. All options are mutually inclusive so you can combine them according to your need. The command output will be the most comprehensive if you combine all options, and you will receive the least details if you use no additional options.

Table 28.2. The abrt-cli list command options

Option	Description
	With no additional option, the abrt-cli list command displays only basic information for problems that have not been reported yet.
-d, --detailed	Displays all stored information about problems listed, including a backtrace if it has already been generated.
-f, --full	Displays basic information for all problems including the already-reported ones.
-v, --verbose	Provides additional information on its actions.

If you want to view information just about one particular problem, you can use the command:

```
abrt-cli info directory
```

...where *directory* stands for the **problem data directory** of the problem that is being viewed. The

following table shows options available with the **abrt-cli info** command. All options are mutually inclusive so you can combine them according to your need. The command output will be the most comprehensive if you combine all options, and you will receive the least details if you use no additional options.

Table 28.3. The abrt-cli info command options

Option	Description
	With no additional option, the abrt-cli info command displays only basic information for the problem specified by the problem data directory argument.
-d, --detailed	Displays all stored information for the problem specified by the problem data directory argument, including a backtrace if it has already been generated.
-v, --verbose	abrt-cli info provides additional information on its actions.

28.3.2. Reporting Problems

To report a certain problem, use the command:

abrt-cli report *directory*

...where *directory* stands for the **problem data directory** of the problem that is being reported. For example:

```
~]$ abrt-cli report /var/spool/abrt/ccpp-2011-09-13-10:18:14-2895
How you would like to analyze the problem?
1) Collect .xsession-errors
2) Local GNU Debugger
Select analyzer: _
```

ABRT prompts you to select an analyzer event for the problem that is being reported. After selecting an event, the problem is analyzed. This can take a considerable amount of time. When the problem report is ready, **abrt-cli** opens a text editor with the content of the report. You can see what is being reported, and you can fill in instructions on how to reproduce the crash and other comments. You should also check the backtrace, because the backtrace might be sent to a public server and viewed by anyone, depending on the problem reporter event settings.



NOTE

You can choose which text editor is used to check the reports. **abrt-cli** uses the editor defined in the **ABRT_EDITOR** environment variable. If the variable is not defined, it checks the **VISUAL** and **EDITOR** variables. If none of these variables is set, **vi** is used. You can set the preferred editor in your **.bashrc** configuration file. For example, if you prefer GNU Emacs, add the following line to the file:

```
export VISUAL=emacs
```

When you are done with the report, save your changes and close the editor. You will be asked which of the configured **ABRT** reporter events you want to use to send the report.

- How would you like to report the problem?
- 1) Logger
 - 2) Red Hat Customer Support
- Select reporter(s): _

After selecting a reporting method, you can proceed with reviewing data to be sent with the report. The following table shows options available with the **abrt-cli report** command.

Table 28.4. The abrt-cli report command options

Option	Description
	With no additional option, the abrt-cli report command provides the usual output.
-v, --verbose	abrt-cli report provides additional information on its actions.

28.3.3. Deleting Problems

If you are certain that you do not want to report a particular problem, you can delete it. To delete a problem so **ABRT** does not keep information about it, use the command:

abrt-cli rm *directory*

...where *directory* stands for the problem data directory of the problem being deleted. For example:

```
~]$ abrt-cli rm /var/spool/abrt/ccpp-2011-09-12-18:37:24-4413
rm '/var/spool/abrt/ccpp-2011-09-12-18:37:24-4413'
```



NOTE

Note that **ABRT** performs a detection of duplicate problems by comparing new problems with all locally saved problems. For a repeating crash, **ABRT** requires you to act upon it only once. However, if you delete the crash dump of that problem, the next time this specific problem occurs, **ABRT** will treat it as a new crash: **ABRT** will alert you about it, prompt you to fill in a description, and report it. To avoid having **ABRT** notifying you about a recurring problem, do not delete its problem data.

The following table shows options available with the **abrt-cli rm** command.

Table 28.5. The abrt-cli rm command options

Option	Description
	With no additional option, the abrt-cli rm command removes the specified problem data directory with all its contents.

Option	Description
-v, --verbose	abrt-cli rm provides additional information on its actions.

28.4. CONFIGURING ABRT

A *problem* life cycle is driven by *events* in **ABRT**. For example:

- Event 1 – a problem data directory is created.
- Event 2 – problem data is analyzed.
- Event 3 – a problem is reported to Bugzilla.

When a problem is detected and its defining data is stored, the problem is processed by running events on the problem's data directory. For more information on events and how to define one, see [Section 28.4.1, "ABRT Events"](#). Standard **ABRT** installation currently supports several default events that can be selected and used during problem reporting process. See [Section 28.4.2, "Standard ABRT Installation Supported Events"](#) to see the list of these events.

Upon installation, **ABRT** and **libreport** place their respective configuration files into the several directories on a system:

- **/etc/libreport/** – contains the **report_event.conf** main configuration file. More information about this configuration file can be found in [Section 28.4.1, "ABRT Events"](#).
- **/etc/libreport/events/** – holds files specifying the default setting of predefined events.
- **/etc/libreport/events.d/** – keeps configuration files defining events.
- **/etc/libreport/plugins/** – contains configuration files of programs that take part in events.
- **/etc/abrt/** – holds **ABRT** specific configuration files used to modify the behavior of **ABRT**'s services and programs. More information about certain specific configuration files can be found in [Section 28.4.4, "ABRT Specific Configuration"](#).
- **/etc/abrt/plugins/** – keeps configuration files used to override the default setting of **ABRT**'s services and programs. For more information on some specific configuration files see [Section 28.4.4, "ABRT Specific Configuration"](#).

28.4.1. ABRT Events

Each event is defined by one rule structure in a respective configuration file. The configuration files are typically stored in the **/etc/libreport/events.d/** directory. These configuration files are used by the main configuration file, **/etc/libreport/report_event.conf**.

The **/etc/libreport/report_event.conf** file consists of *include* directives and *rules*. Rules are typically stored in other configuration files in the **/etc/libreport/events.d/** directory. In the standard installation, the **/etc/libreport/report_event.conf** file contains only one include directive:

```
include events.d/*.conf
```

If you would like to modify this file, please note that it respects shell metacharacters (*,\$,?, etc.) and interprets relative paths relatively to its location.

Each *rule* starts with a line with a non-space leading character, all subsequent lines starting with the **space** character or the **tab** character are considered a part of this rule. Each *rule* consists of two parts, a *condition* part and a *program* part. The condition part contains conditions in one of the following forms:

- *VAR=VAL*,
- *VAR!=VAL*, or
- *VAL~=*REGEX**

...where:

- *VAR* is either the **EVENT** key word or a name of a problem data directory element (such as **executable**, **package**, **hostname**, etc.),
- *VAL* is either a name of an event or a problem data element, and
- *REGEX* is a regular expression.

The program part consists of program names and shell interpretable code. If all conditions in the condition part are valid, the program part is run in the shell. The following is an event example:

```
EVENT=post-create date > /tmp/dt
echo $HOSTNAME `uname -r`
```

This event would overwrite the contents of the **/tmp/dt** file with the current date and time, and print the host name of the machine and its kernel version on the standard output.

Here is an example of a yet more complex event which is actually one of the predefined events. It saves relevant lines from the **~/.xsession-errors** file to the problem report for any problem for which the **abrt-ccpp** services has been used to process that problem, and the crashed application has loaded any X11 libraries at the time of crash:

```
EVENT=analyze_xsession_errors analyzer=CCpp dso_list~=.*/libX11.*
test -f ~/.xsession-errors || { echo "No ~/.xsession-errors"; exit 1; }
test -r ~/.xsession-errors || { echo "Can't read ~/.xsession-errors"; exit 1; }
executable=`cat executable` &&
base_executable=${executable##*/} &&
grep -F -e "$base_executable" ~/.xsession-errors | tail -999 >xsession_errors &&
echo "Element 'xsession_errors' saved"
```

The set of possible events is not hard-set. System administrators can add events according to their need. Currently, the following event names are provided with standard **ABRT** and **libreport** installation:

post-create

This event is run by **abrtd** on newly created problem data directories. When the **post-create** event is run, **abrtd** checks whether the UUID identifier of the new problem data matches the UUID of any already existing problem directories. If such a problem directory exists, the new problem data is deleted.

analyze_name_suffix

...where *name_suffix* is the adjustable part of the event name. This event is used to process collected data. For example, the **analyze_LocalGDB** runs the GNU Debugger (**GDB**) utility on a core dump of an application and produces a backtrace of a program. You can view the list of analyze events and choose from it using **abrt-gui**.

collect_name_suffix

...where *name_suffix* is the adjustable part of the event name. This event is used to collect additional information on a problem. You can view the list of collect events and choose from it using **abrt-gui**.

report_name_suffix

...where *name_suffix* is the adjustable part of the event name. This event is used to report a problem. You can view the list of report events and choose from it using **abrt-gui**.

Additional information about events (such as their description, names and types of parameters which can be passed to them as environment variables, and other properties) is stored in the **/etc/libreport/events/event_name.xml** files. These files are used by **abrt-gui** and **abrt-cli** to make the user interface more friendly. Do not edit these files unless you want to modify the standard installation.

28.4.2. Standard ABRT Installation Supported Events

Standard **ABRT** installation currently provides a number of default analyzing, collecting and reporting events. Some of these events are also configurable using the **ABRT** GUI application (for more information on event configuration using **ABRT** GUI, see [Section 28.4.3, “Event Configuration in ABRT GUI”](#)). **ABRT** GUI only shows the event's unique part of the name which is more readable the user, instead of the complete event name. For example, the **analyze_xsession_errors** event is shown as **Collect .xsession-errors** in **ABRT** GUI. The following is a list of default analyzing, collecting and reporting events provided by the standard installation of **ABRT**:

analyze_VMcore – Analyze VM core

Runs **GDB** (the GNU debugger) on problem data of an application and generates a **backtrace** of the kernel. It is defined in the **/etc/libreport/events.d/vmcore_event.conf** configuration file.

analyze_LocalGDB – Local GNU Debugger

Runs **GDB** (the GNU debugger) on problem data of an application and generates a **backtrace** of a program. It is defined in the **/etc/libreport/events.d/ccpp_event.conf** configuration file.

analyze_xsession_errors – Collect .xsession-errors

Saves relevant lines from the **~/.xsession-errors** file to the problem report. It is defined in the **/etc/libreport/events.d/ccpp_event.conf** configuration file.

report_Logger – Logger

Creates a problem report and saves it to a specified local file. It is defined in the **/etc/libreport/events.d/print_event.conf** configuration file.

report_RHTSupport – Red Hat Customer Support

Reports problems to the Red Hat Technical Support system. This possibility is intended for users of Red Hat Enterprise Linux. It is defined in the **/etc/libreport/events.d/rhtsupport_event.conf** configuration file.

report_Mailx – Mailx

Sends a problem report via the **Mailx** utility to a specified email address. It is defined in the **/etc/libreport/events.d/mailx_event.conf** configuration file.

report_Kerneloops – Kerneloops.org

Sends a kernel problem to the oops tracker. It is defined in the **/etc/libreport/events.d/koops_event.conf** configuration file.

report_Uploader – Report uploader

Uploads a tarball (.tar.gz) archive with problem data to the chosen destination using the **FTP** or the **SCP** protocol. It is defined in the **/etc/libreport/events.d/uploader_event.conf** configuration file.

28.4.3. Event Configuration in ABRT GUI

Events can use parameters passed to them as environment variables (for example, the **report_Logger** event accepts an output file name as a parameter). Using the respective **/etc/libreport/events/event_name.xml** file, ABRT GUI determines which parameters can be specified for a selected event and allows a user to set the values for these parameters. These values are saved by ABRT GUI and reused on subsequent invocations of these events.

Open the **Event Configuration** window by clicking **Edit → Preferences**. This window shows a list of all available events that can be selected during the reporting process. When you select one of the configurable events, you can click the **Configure Event** button and you will be able to configure settings for that event. If you change any of the events' parameters, they are saved in the **Gnome** keyring and will be used in the future GUI sessions.



NOTE

All files in the **/etc/libreport/** directory hierarchy are world readable and are meant to be used as global settings. Thus, it is not advisable to store user names, passwords or any other sensitive data in them. The per-user settings (set in the GUI application and readable by the owner of **\$HOME** only) are stored in the **Gnome** keyring or can be stored in a text file in **\$HOME/.abrt/*.conf** for use in **abrt-cli**.

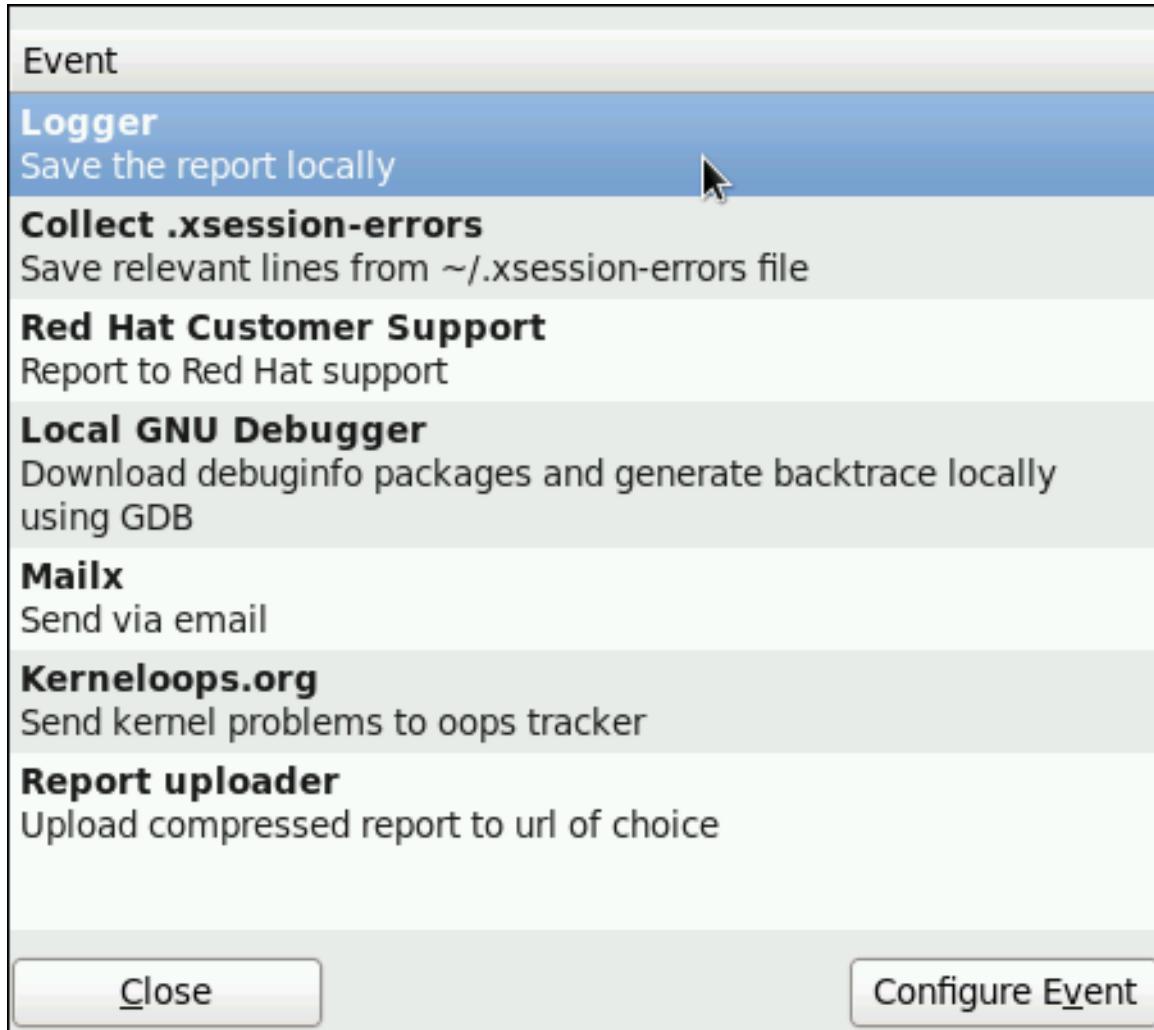


Figure 28.12. The Event Configuration Window

The following is a list of all configuration options available for each predefined event that is configurable in the **ABRT** GUI application.

Logger

In the **Logger** event configuration window, you can configure the following parameter:

- **Log file** – Specifies a file into which the crash reports are saved (by default, set to `/var/log/abrt.log`).

When the **Append** option is checked, the **Logger** event will append new crash reports to the log file specified in the **Logger file** option. When unchecked, the new crash report always replaces the previous one.

Red Hat Customer Support

In the **Red Hat Customer Support** event configuration window, you can configure the following parameters:

- **RH Portal URL** – Specifies the Red Hat Customer Support URL where crash dumps are sent (by default, set to <https://api.access.redhat.com/rs>).
- **Username** – User login which is used to log into Red Hat Customer Support and create a Red Hat Customer Support database entry for a reported crash. Use your *Red Hat Login* acquired by creating an account on <https://www.redhat.com/en>, the Red Hat Customer

Portal (<https://access.redhat.com/home>) or the Red Hat Network (<https://rhn.redhat.com/>).

- **Password** – Password used to log into Red Hat Customer Support (that is, password associated with your *Red Hat Login*)

When the **SSL verify** option is checked, the **SSL** protocol is used when sending the data over the network.

MailX

In the **MailX** event configuration window, you can configure the following parameters:

- **Subject** – A string that appears in the **Subject** field of a problem report email sent by **MailX** (by default, set to "**[abrt] detected a crash**").
- **Sender** – A string that appears in the **From** field of a problem report email.
- **Recipient** – Email address of the recipient of a problem report email.

When the **Send Binary Data** option is checked, the problem report email will also contain all binary files associated with the problem in an attachment. The core dump file is also sent as an attachment.

Kerneloops.org

In the **Kerneloops.org** event configuration window, you can configure the following parameter:

- **Kerneloops URL** – Specifies the URL where Kernel problems are reported to (by default, set to <http://submit.kerneloops.org/submitoops.php>)

Report Uploader

In the **Report Uploader** event configuration widow, you can configure the following parameter:

- **URL** – Specifies the URL where a tarball containing compressed problem data is uploaded using the **FTP** or **SCP** protocol (by default, set to <ftp://localhost:/tmp/upload>).

28.4.4. ABRT Specific Configuration

Standard **ABRT** installation currently provides the following **ABRT** specific configuration files:

- **/etc/abrt/abrt.conf** – allows you to modify the behavior of the **abrt** service.
- **/etc/abrt/abrt-action-save-package-data.conf** – allows you to modify the behavior of the **abrt-action-save-package-data** program.
- **/etc/abrt/plugins/CCpp.conf** – allows you to modify the behavior of **ABRT**'s core catching hook.

The following configuration directives are supported in the **/etc/abrt/abrt.conf** file:

WatchCrashdumpArchiveDir = /var/spool/abrt-upload

This directive is commented out by default. Enable it if you want **abrt** to auto-unpack crashdump tarball archives (.tar.gz) which are located in the specified directory. In the example above, it is the **/var/spool/abrt-upload** directory. Whichever directory you specify in this directive, you must ensure

that it exists and it is writable for **abrt**. The **ABRT** daemon will not create it automatically. If you change the default value of this option, be aware that in order to ensure proper functionality of **ABRT**, this directory **must not** be the same as the directory specified for the **DumpLocation** option.



WARNING

Changing the location for crashdump archives will cause SELinux denials unless you reflect the change in respective SELinux rules first. See the **abrt_selinux(8)** manual page for more information on running **ABRT** in SELinux.

Remember that if you enable this option when using SELinux, you need to execute the following command in order to set the appropriate Boolean allowing **ABRT** to write into the `public_content_rw_t` domain:

```
setsebool -P abrt_anon_write 1
```

MaxCrashReportsSize = size_in_megabytes

This option sets the amount of storage space, in megabytes, used by **ABRT** to store all problem information from all users. The default setting is **1000** MB. Once the quota specified here has been met, **ABRT** will continue catching problems, and in order to make room for the new crash dumps, it will delete the oldest and largest ones.

DumpLocation = /var/spool/abrt

This directive is commented out by default. It specifies the location where problem data directories are created and in which problem core dumps and all other problem data are stored. The default location is set to the **/var/spool/abrt** directory. Whichever directory you specify in this directive, you must ensure that it exists and it is writable for **abrt**. If you change the default value of this option, be aware that in order to ensure proper functionality of **ABRT**, this directory **must not** be the same as the directory specified for the **WatchCrashdumpArchiveDir** option.



WARNING

Changing the dump location will cause SELinux denials unless you reflect the change in respective SELinux rules first. See the **abrt_selinux(8)** manual page for more information on running **ABRT** in SELinux.

Remember that if you enable this option when using SELinux, you need to execute the following command in order to set the appropriate Boolean allowing **ABRT** to write into the `public_content_rw_t` domain:

```
setsebool -P abrt_anon_write 1
```

The following configuration directives are supported in the **/etc/abrt/abrt-action-save-package-data.conf** file:

OpenGPGCheck = yes/no

Setting the **OpenGPGCheck** directive to **yes** (the default setting) tells **ABRT** to *only* analyze and handle crashes in applications provided by packages which are signed by the GPG keys whose locations are listed in the **/etc/abrt/gpg_keys** file. Setting **OpenGPGCheck** to **no** tells **ABRT** to catch crashes in all programs.

BlackList = nspluginwrapper, valgrind, strace, [more_packages]

Crashes in packages and binaries listed after the **BlackList** directive will not be handled by **ABRT**. If you want **ABRT** to ignore other packages and binaries, list them here separated by commas.

ProcessUnpackaged = yes/no

This directive tells **ABRT** whether to process crashes in executables that do not belong to any package. The default setting is **no**.

BlackListedPaths = /usr/share/doc/*, */example*

Crashes in executables in these paths will be ignored by **ABRT**.

The following configuration directives are supported in the **/etc/abrt/plugins/CCpp.conf** file:

MakeCompatCore = yes/no

This directive specifies whether **ABRT**'s core catching hook should create a core file, as it could be done if **ABRT** would not be installed. The core file is typically created in the current directory of the crashed program but only if the **ulimit -c** setting allows it. The directive is set to **yes** by default.

SaveBinaryImage = yes/no

This directive specifies whether **ABRT**'s core catching hook should save a binary image to a core dump. It is useful when debugging crashes which occurred in binaries that were deleted. The default setting is **no**.

28.4.5. Configuring ABRT to Detect a Kernel Panic

With Red Hat Enterprise Linux 6.3, **ABRT** can detect a kernel panic using the **abrt-vmcore** service, which is provided by the abrt-addon-vmcore package. The service starts automatically on system boot and searches for a core dump file in the **/var/crash/** directory. If a core dump file is found, **abrt-vmcore** creates the **problem data directory** in the **/var/spool/abrt/** directory and moves the core dump file to the newly created problem data directory. After the **/var/crash/** directory is searched through, the service is stopped until the next system boot.

To configure **ABRT** to detect a kernel panic, perform the following steps:

1. Ensure that the **kdump** service is enabled on the system. Especially, the amount of memory that is reserved for the kdump kernel has to be set correctly. You can set it by using the **system-config-kdump** graphical tool, or by specifying the **crashkernel** parameter in the list of kernel options in the **/etc/grub.conf** configuration file. See [Chapter 32, The kdump Crash Recovery Service](#) for details on how to enable and configure **kdump**.
2. Install the abrt-addon-vmcore package using the **Yum** package installer:

```
~]# yum install abrt-addon-vmcore
```

This installs the **abrt-vmcore** service with respective support and configuration files. Please note that the `abrt-addon-vmcore` package is provided by the Optional subscription channel. See [Section 8.4.8, "Adding the Optional and Supplementary Repositories"](#) for more information on Red Hat additional channels.

3. Reboot the system for the changes to take effect.

Unless **ABRT** is configured differently, problem data for any detected kernel panic is now stored in the `/var/spool/abrt/` directory and can be further processed by **ABRT** just as any other detected kernel oops.

28.4.6. Automatic Downloads and Installation of Debuginfo Packages

ABRT can be configured to automatically download and install packages needed for debugging of particular problems. This feature can be useful if you want to debug problems locally in your company environment. To enable automatic debuginfo downloads and installation, ensure that your system fulfills the following conditions:

- The `/etc/libreport/events.d/ccpp_event.conf` file contains the following analyzer event, which is present uncommented in default configuration:

```
EVENT=analyze_LocalGDB analyzer=CCpp
abrt-action-analyze-core --core=coredump -o build_ids &&
# In RHEL we don't want to install anything by default
# and also this would fail, as the debuginfo repositories.
# are not available without root password rhbz#759443
#/usr/libexec/abrt-action-install-debuginfo-to-abrt-cache --size_mb=4096 &&
abrt-action-generate-backtrace &&
abrt-action-analyze-backtrace
```

- The `/etc/libreport/events.d/ccpp_event.conf` file contains the following line, which allows ABRT to run binary to install debuginfo packages for the problems being analyzed. This line is, in order to avoid installations of unnecessary content, commented out by default so you have to remove the leading `#` character to enable it:

```
/usr/libexec/abrt-action-install-debuginfo-to-abrt-cache --size_mb=4096 &&
```

- The `gdb` package, which allows you to generate a backtrace during a problem analysis, is installed on your system. If needed, see [Section 8.2.4, "Installing Packages"](#) for more information on how to install packages with the `Yum` package manager.



IMPORTANT

Note that debuginfo packages are installed using the `rhnplugin` which requires root privileges. Therefore, you have to run ABRT as `root` to be able to install debuginfo packages.

28.4.7. Configuring Automatic Reporting for Specific Types of Crashes

ABRT can be configured to report any detected issues or crashes automatically without any user interaction. This can be achieved by specifying an analyze-and-report rule as a *post-create* rule. For

example, you can instruct ABRT to report Python crashes to Bugzilla immediately without any user interaction by enabling the rule and replacing the **EVENT=report_Bugzilla** condition with the **EVENT=post-create** condition in the **/etc/libreport/events.d/python_event.conf** file. The new rule will look like the follows:

```
EVENT=post-create analyzer=Python
test -f component || abrt-action-save-package-data
reporter-bugzilla -c /etc/abrt/plugins/bugzilla.conf
```



WARNING

Please note that the **post-create** event is run by **abrt**, which usually runs with root privileges.

28.4.8. Uploading and Reporting Using a Proxy Server

The **reporter-bugzilla** and the **reporter-upload** tools respect the **http_proxy** and the **ftp_proxy** environment variables. When you use environment variables as a part of a reporting event, they inherit their values from the process which performs reporting, usually **abrt-gui** or **abrt-cli**. Therefore, you can specify **HTTP** or **FTP** proxy servers by using these variables in your working environment.

If you arrange these tools to be a part of the **post-create** event, they will run as children of the **abrt** process. You should either adjust the environment of abrt or modify the rules to set these variables. For example:

```
EVENT=post-create analyzer=Python
test -f component || abrt-action-save-package-data
export http_proxy=http://proxy.server:8888/
reporter-bugzilla -c /etc/abrt/plugins/bugzilla.conf
```

28.4.9. Configuring Automatic Reporting

ABRT can be configured to use *μReports*. This additional type of bug report has these advantages:

- Once enabled, *μReports* are sent *automatically*, without user interaction. In contrast, the normal reports are not sent until manually triggered by the user.
- μReports* are *anonymous and do not contain sensitive information*. This eliminates the risk that unwanted data will be submitted automatically.
- A *μReport* represents the detected problem as a JSON object. Therefore, it is machine-readable and can be created and processed automatically.
- μReports* are smaller than full bug reports.
- μReports* do not require downloading large amounts of debugging information.

μReports serve several goals. They help to prevent duplicate customer cases that might get created because of multiple occurrences of the same bug. Additionally, *μReports* enable gathering statistics of bug occurrences and finding known bugs across different systems. Finally, if *authenticated μReports* are

enabled as described at the end of this section, ABRT can automatically present instant solutions to the customers. However, µReports do not necessarily provide engineers with enough information to fix the bug, for which a full bug report may be necessary.

A µReport generally contains the following information:

- a call stack trace of a program without any variables, or, in case of multi-threaded C, C++, and Java programs, multiple stack traces
- which operating system is used
- versions of the RPM packages involved in the crash
- whether the program ran under the **root** user
- for kernel oops, possibly information about host hardware



WARNING

Do not enable µReports if you do not want to share information about your hardware with Red Hat.

For µReport examples, see the [Examples of µReports](#) article.

With µReports enabled, the following happens by default when a crash is detected:

1. ABRT submits a µReport with basic information about the problem to Red Hat's ABRT server.
2. The server determines whether the problem is already in the bug database.
3. If it is, the server returns a short description of the problem along with a URL of the reported case.

If not, the server invites the user to submit a full problem report.

To enable µReports for all users, run as **root**:

```
~]# abrt-auto-reporting enabled
```

or add the following line to the **/etc/abrt/abrt.conf** file:

```
AutoreportingEnabled = yes
```

User-specific configuration is located in the **\$USER/.config/abrt** directory. It overrides the system-wide configuration.

To apply the new configuration, restart the ABRT services by running:

```
~]# service abrtd restart
```

The default autoreporting behavior – sending μReports – can be changed. To do that, assign a different ABRT event to the **AutoreportingEvent** directive in the **/etc/abrt/abrt.conf** configuration file. See [Section 28.4.2, “Standard ABRT Installation Supported Events”](#) for an overview of the standard events.

In Red Hat Enterprise Linux 7.1 and later, customers can also send *authenticated* μReports, which contain more information: hostname, machine-id (taken from the **/etc/machine-id** file), and RHN account number. The advantage of authenticated μReports is that they go directly to the Red Hat Customer Portal, and not only to Red Hat's private crash-report server, as the regular μReports do. This enables Red Hat to provide customers with instant solutions to crashes.

To turn the *authenticated* automatic reporting on, run the following command as **root**:

```
~]# abrt-auto-reporting enabled -u RHN_username
```

Replace *RHN_username* with your Red Hat Network username. This command will ask for your password and save it in plain text into the **/etc/libreport/plugins/rhtsupport.conf** file.

28.5. CONFIGURING CENTRALIZED CRASH COLLECTION

You can set up **ABRT** so that crash reports are collected from multiple systems and sent to a dedicated system for further processing. This is useful when an administrator does not want to log into hundreds of systems and manually check for crashes found by **ABRT**. In order to use this method, you need to install the **libreport-plugin-reportuploader** plug-in (**yum install libreport-plugin-reportuploader**). See the following sections on how to configure systems to use ABRT's centralized crash collection.

28.5.1. Configuration Steps Required on a Dedicated System

Complete the following steps on a dedicated (server) system:

1. Create a directory to which you want the crash reports to be uploaded to. Usually, **/var/spool/abrt-upload/** is used (the rest of the document assumes you are using this directory). Make sure this directory is writable by the abrt user.



NOTE

When the abrt-desktop package is installed, it creates a new system user and a group, both named **abrt**. This user is used by the **abrtd** daemon, for example, as the owner:group of **/var/spool/abrt/*** directories.

2. In the **/etc/abrt/abrt.conf** configuration file, set the **WatchCrashdumpArchiveDir** directive to the following:

```
WatchCrashdumpArchiveDir = /var/spool/abrt-upload/
```

3. Choose your preferred upload mechanism; for example, **FTP** or **SCP**. For more information on how to configure **FTP**, see [Section 21.2, “FTP”](#). For more information on how to configure **SCP**, see [Section 14.4.2, “Using the scp Utility”](#).

It is advisable to check whether your upload method works. For example, if you use **FTP**, upload a file using an interactive **FTP** client:

```
~]$ ftp
ftp> open servername
```

```
Name: username
Password: password
ftp> cd /var/spool/abrt-upload
250 Operation successful
ftp> put testfile
ftp> quit
```

Check whether ***testfile*** appeared in the correct directory on the server system.

4. The **MaxCrashReportsSize** directive (in the **/etc/abrt/abrt.conf** configuration file) needs to be set to a larger value if the expected volume of crash data is larger than the default **1000 MB**.
5. Consider whether you would like to generate a backtrace of C/C++ crashes.

You can disable backtrace generation on the server if you do not want to generate backtraces at all, or if you decide to create them locally on the machine where a problem occurred. In the standard ABRT installation, a backtrace of a C/C++ crash is generated using the following rule in the **/etc/libreport/events.d/ccpp_events.conf** configuration file:

```
EVENT=analyze_LocalGDB analyzer=CCpp
abrt-action-analyze-core.py --core=coredump -o build_ids &&
abrt-action-install-debuginfo-to-abrt-cache --size_mb=4096 &&
abrt-action-generate-backtrace &&
abrt-action-analyze-backtrace
```

You can ensure that this rule is not applied for uploaded problem data by adding the **remote!=1** condition to the rule.

6. Decide whether you want to collect package information (the **package** and the **component** elements) in the problem data. See [Section 28.5.3, “Saving Package Information”](#) to find out whether you need to collect package information in your centralized crash collection configuration and how to configure it properly.

28.5.2. Configuration Steps Required on a Client System

Complete the following steps on every client system which will use the central management method:

1. If you do not want to generate a backtrace, or if you decided to generate it on a server system, you need to delete or comment out the corresponding rules in the **/etc/libreport/events.d/ccpp_events.conf** file. See [Section 28.5.1, “Configuration Steps Required on a Dedicated System”](#) for an example of such a example.
2. If you decided to not collect package information on client machines, delete, comment out or modify the rule which runs abrt-action-save-package-data in the **/etc/libreport/events.d/abrt_event.conf** file. See [Section 28.5.3, “Saving Package Information”](#) to find out whether you need to collect package information in your centralized crash collection configuration and how to configure it properly.
3. Add a rule for uploading problem reports to the server system in the corresponding configuration file. For example, if you want to upload all problems automatically as soon as they are detected, you can use the following rule in the **/etc/libreport/events.d/abrt_event.conf** configuration file:

```
EVENT=post-create
reporter-upload -u scp://user:password@server.name/directory
```

Alternatively, you can use a similar rule that runs the reporter-upload program as the **report_SFX** event if you want to store problem data locally on clients and upload it later using ABRT GUI/CLI. The following is an example of such an event:

```
EVENT=report_UploadToMyServer
reporter-upload -u scp://user:password@server.name/directory
```

28.5.3. Saving Package Information

In a single-machine **ABRT** installation, problems are usually reported to external bug databases such as RHTSupport or Bugzilla. Reporting to these bug databases usually requires knowledge about the component and package in which the problem occurred. The **post-create** event runs the **abrt-action-save-package-data** tool (among other steps) in order to provide this information in the standard **ABRT** installation.

If you are setting up a centralized crash collection system, your requirements may be significantly different. Depending on your needs, you have two options:

Internal analysis of problems

After collecting problem data, you do not need to collect package information if you plan to analyze problems in-house, without reporting them to any external bug databases. You might be also interested in collecting crashes that occur in programs written by your organization or third-party applications installed on your system. If such a program is a part of an RPM package, then on *client systems* and a *dedicated crash collecting system*, you can only add the respective GPG key to the **/etc/abrt/gpg_keys** file or set the following line in the **/etc/abrt/abrt-action-save-package-data.conf** file:

```
OpenGPGCheck = no
```

If the program does not belong to any RPM package, take the following steps on both, *client systems* and a *dedicated crash collecting system*:

- Remove the following rule from the **/etc/libreport/events.d/abrt_event.conf** file:

```
EVENT=post-create component=
abrt-action-save-package-data
```

- Prevent deletion of problem data directories which do not correspond to any installed package by setting the following directive in the **/etc/abrt/abrt-action-save-package-data.conf** file:

```
ProcessUnpackaged = yes
```

Reporting to external bug database

Alternatively, you may want to report crashes to RHTSupport or Bugzilla. In this case, you need to collect package information. Generally, client machines and dedicated crash collecting systems have non-identical sets of installed packages. Therefore, it may happen that problem data uploaded from a client does not correspond to any package installed on the dedicated crash collecting system. In the standard **ABRT** configuration, this will lead to deletion of problem data (ABRT will consider it to be a crash in an unpackaged executable). To prevent this from happening, it is necessary to modify **ABRT**'s configuration on the *dedicated system* in the following way:

- Prevent inadvertent collection of package information for problem data uploaded from client machines, by adding the **remote!=1** condition in the **/etc/libreport/events.d/abrt_event.conf** file:

```
EVENT=post-create remote!=1 component=
abrt-action-save-package-data
```

- Prevent deletion of problem data directories which do not correspond to any installed package by setting the following directive in **/etc/abrt/abrt-action-save-package-data.conf**:

```
ProcessUnpackaged = yes
```



NOTE

Note that in this case, no such modifications are necessary on client systems: they continue to collect package information, and continue to ignore crashes in unpackaged executables.

28.5.4. Testing ABRT's Crash Detection

After completing all the steps of the configuration process, the basic setup is finished. To test that this setup works properly use the **kill -s SEGV PID** command to terminate a process on a client system. For example, start a **sleep** process and terminate it with the **kill** command in the following way:

```
~]$ sleep 100 &
[1] 2823
~]$ kill -s SIGSEGV 2823
```

ABRT should detect a crash shortly after executing the **kill** command. Check that the crash was detected by **ABRT** on the client system (this can be checked by examining the appropriate syslog file, by running the **abrt-cli list --full** command, or by examining the crash dump created in the **/var/spool/abrt** directory), copied to the server system, unpacked on the server system and can be seen and acted upon using **abrt-cli** or **abrt-gui** on the server system.

CHAPTER 29. OPROFILE

OProfile is a low overhead, system-wide performance monitoring tool. It uses the performance monitoring hardware on the processor to retrieve information about the kernel and executables on the system, such as when memory is referenced, the number of L2 cache requests, and the number of hardware interrupts received. On a Red Hat Enterprise Linux system, the oprofile package must be installed to use this tool.

Many processors include dedicated performance monitoring hardware. This hardware makes it possible to detect when certain events happen (such as the requested data not being in cache). The hardware normally takes the form of one or more *counters* that are incremented each time an event takes place. When the counter value, essentially rolls over, an interrupt is generated, making it possible to control the amount of detail (and therefore, overhead) produced by performance monitoring.

OProfile uses this hardware (or a timer-based substitute in cases where performance monitoring hardware is not present) to collect *samples* of performance-related data each time a counter generates an interrupt. These samples are periodically written out to disk; later, the data contained in these samples can then be used to generate reports on system-level and application-level performance.

OProfile is a useful tool, but be aware of some limitations when using it:

- *Use of shared libraries* – Samples for code in shared libraries are not attributed to the particular application unless the **--separate=library** option is used.
- *Performance monitoring samples are inexact* – When a performance monitoring register triggers a sample, the interrupt handling is not precise like a divide by zero exception. Due to the out-of-order execution of instructions by the processor, the sample may be recorded on a nearby instruction.
- ***opreport*** does not associate samples for inline functions properly – **opreport** uses a simple address range mechanism to determine which function an address is in. Inline function samples are not attributed to the inline function but rather to the function the inline function was inserted into.
- *OProfile accumulates data from multiple runs* – OProfile is a system-wide profiler and expects processes to start up and shut down multiple times. Thus, samples from multiple runs accumulate. Use the command **opcontrol --reset** to clear out the samples from previous runs.
- *Hardware performance counters do not work on guest virtual machines* – Because the hardware performance counters are not available on virtual systems, you need to use the **timer** mode. Run the command **opcontrol --deinit**, and then execute **modprobe oprofile timer=1** to enable the **timer** mode.
- *Non-CPU-limited performance problems* – OProfile is oriented to finding problems with CPU-limited processes. OProfile does not identify processes that are asleep because they are waiting on locks or for some other event to occur (for example an I/O device to finish an operation).

29.1. OVERVIEW OF TOOLS

[Table 29.1, “OProfile Commands”](#) provides a brief overview of the tools provided with the `oprofile` package.

Table 29.1. OProfile Commands

Command	Description
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 29.5.4, “Using opannotate” for details.
opcontrol	Configures what data is collected. See Section 29.2, “Configuring OProfile” for details.
opreport	Retrieves profile data. See Section 29.5.1, “Using opreport” for details.
oprofiled	Runs as a daemon to periodically write sample data to disk.

29.2. CONFIGURING OPROFILE

Before OProfile can be run, it must be configured. At a minimum, selecting to monitor the kernel (or selecting not to monitor the kernel) is required. The following sections describe how to use the **opcontrol** utility to configure OProfile. As the **opcontrol** commands are executed, the setup options are saved to the **/root/.oprofile/daemonrc** file.

29.2.1. Specifying the Kernel

First, configure whether OProfile should monitor the kernel. This is the only configuration option that is required before starting OProfile. All others are optional.

To monitor the kernel, execute the following command as root:

```
~]# opcontrol --setup --vmlinux=/usr/lib/debug/lib/modules/`uname -r`/vmlinux
```



IMPORTANT

The debuginfo package for the kernel must be installed (which contains the uncompressed kernel) in order to monitor the kernel.

To configure OProfile not to monitor the kernel, execute the following command as root:

```
~]# opcontrol --setup --no-vmlinux
```

This command also loads the **oprofile** kernel module, if it is not already loaded, and creates the **/dev/oprofile/** directory, if it does not already exist. See [Section 29.6, “Understanding /dev/oprofile/”](#) for details about this directory.

Setting whether samples should be collected within the kernel only changes what data is collected, not how or where the collected data is stored. To generate different sample files for the kernel and application libraries, see [Section 29.2.3, “Separating Kernel and User-space Profiles”](#).

29.2.2. Setting Events to Monitor

Most processors contain *counters*, which are used by OProfile to monitor specific events. As shown in [Table 29.2, “OProfile Processors and Counters”](#), the number of counters available depends on the processor.

Table 29.2. OProfile Processors and Counters

Processor	cpu_type	Number of Counters
AMD64	x86-64/hammer	4
AMD Athlon	i386/athlon	4
AMD Family 10h	x86-64/family10	4
AMD Family 11h	x86-64/family11	4
AMD Family 12h	x86-64/family12	4
AMD Family 14h	x86-64/family14	4
AMD Family 15h	x86-64/family15	6
IBM eServer System i and IBM eServer System p	timer	1
IBM POWER4	ppc64/power4	8
IBM POWER5	ppc64/power5	6
IBM PowerPC 970	ppc64/970	8
IBM S/390 and IBM System z	timer	1
Intel Core i7	i386/core_i7	4
Intel Nehalem microarchitecture	i386/nehalem	4
Intel Pentium 4 (non-hyper-threaded)	i386/p4	8
Intel Pentium 4 (hyper-threaded)	i386/p4-ht	4
Intel Westmere microarchitecture	i386/westmere	4

Processor	cpu_type	Number of Counters
TIMER_INT	timer	1

Use [Table 29.2, “OProfile Processors and Counters”](#) to verify that the correct processor type was detected and to determine the number of events that can be monitored simultaneously. **timer** is used as the processor type if the processor does not have supported performance monitoring hardware.

If **timer** is used, events cannot be set for any processor because the hardware does not have support for hardware performance counters. Instead, the timer interrupt is used for profiling.

If **timer** is not used as the processor type, the events monitored can be changed, and counter 0 for the processor is set to a time-based event by default. If more than one counter exists on the processor, the counters other than counter 0 are not set to an event by default. The default events monitored are shown in [Table 29.3, “Default Events”](#).

Table 29.3. Default Events

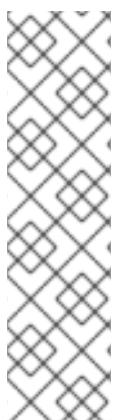
Processor	Default Event for Counter	Description
AMD Athlon and AMD64	CPU_CLK_UNHALTED	The processor's clock is not halted
AMD Family 10h, AMD Family 11h, AMD Family 12h	CPU_CLK_UNHALTED	The processor's clock is not halted
AMD Family 14h, AMD Family 15h	CPU_CLK_UNHALTED	The processor's clock is not halted
IBM POWER4	CYCLES	Processor Cycles
IBM POWER5	CYCLES	Processor Cycles
IBM PowerPC 970	CYCLES	Processor Cycles
Intel Core i7	CPU_CLK_UNHALTED	The processor's clock is not halted
Intel Nehalem microarchitecture	CPU_CLK_UNHALTED	The processor's clock is not halted
Intel Pentium 4 (hyper-threaded and non-hyper-threaded)	GLOBAL_POWER_EVENTS	The time during which the processor is not stopped
Intel Westmere microarchitecture	CPU_CLK_UNHALTED	The processor's clock is not halted
TIMER_INT	(none)	Sample for each timer interrupt

The number of events that can be monitored at one time is determined by the number of counters for the processor. However, it is not a one-to-one correlation; on some processors, certain events must be mapped to specific counters. To determine the number of counters available, execute the following command:

```
~]# ls -d /dev/oprofile/[0-9]*
```

The events available vary depending on the processor type. To determine the events available for profiling, execute the following command as root (the list is specific to the system's processor type):

```
~]# ophelp
```



NOTE

Unless OProfile is properly configured, the **ophelp** fails with the following error message:

Unable to open cpu_type file for reading
Make sure you have done opcontrol --init
cpu_type 'unset' is not valid
you should upgrade oprofile or force the use of timer mode

To configure OProfile, follow the instructions in [Section 29.2, "Configuring OProfile"](#).

The events for each counter can be configured via the command line or with a graphical interface. For more information on the graphical interface, see [Section 29.9, "Graphical Interface"](#). If the counter cannot be set to a specific event, an error message is displayed.

To set the event for each configurable counter via the command line, use **opcontrol**:

```
~]# opcontrol --event=event-name:sample-rate
```

Replace *event-name* with the exact name of the event from **ophelp**, and replace *sample-rate* with the number of events between samples.

29.2.2.1. Sampling Rate

By default, a time-based event set is selected. It creates a sample every 100,000 clock cycles per processor. If the timer interrupt is used, the timer is set to whatever the jiffy rate is and is not user-settable. If the **cpu_type** is not **timer**, each event can have a *sampling rate* set for it. The sampling rate is the number of events between each sample snapshot.

When setting the event for the counter, a sample rate can also be specified:

```
~]# opcontrol --event=event-name:sample-rate
```

Replace *sample-rate* with the number of events to wait before sampling again. The smaller the count, the more frequent the samples. For events that do not happen frequently, a lower count may be needed to capture the event instances.

**WARNING**

Be extremely careful when setting sampling rates. Sampling too frequently can overload the system, causing the system to appear as if it is frozen or causing the system to actually freeze.

29.2.2.2. Unit Masks

Some user performance monitoring events may also require unit masks to further define the event.

Unit masks for each event are listed with the **ophelp** command. The values for each unit mask are listed in hexadecimal format. To specify more than one unit mask, the hexadecimal values must be combined using a bitwise *or* operation.

```
~]# opcontrol --event=event-name:sample-rate:unit-mask
```

29.2.3. Separating Kernel and User-space Profiles

By default, kernel mode and user mode information is gathered for each event. To configure OProfile to ignore events in kernel mode for a specific counter, execute the following command:

```
~]# opcontrol --event=event-name:sample-rate:unit-mask:0
```

Execute the following command to start profiling kernel mode for the counter again:

```
~]# opcontrol --event=event-name:sample-rate:unit-mask:1
```

To configure OProfile to ignore events in user mode for a specific counter, execute the following command:

```
~]# opcontrol --event=event-name:sample-rate:unit-mask:kernel:0
```

Execute the following command to start profiling user mode for the counter again:

```
~]# opcontrol --event=event-name:sample-rate:unit-mask:kernel:1
```

When the OProfile daemon writes the profile data to sample files, it can separate the kernel and library profile data into separate sample files. To configure how the daemon writes to sample files, execute the following command as root:

```
~]# opcontrol --separate=choice
```

choice can be one of the following:

- **none** – Do not separate the profiles (default).
- **library** – Generate per-application profiles for libraries.

- **kernel** – Generate per-application profiles for the kernel and kernel modules.
- **all** – Generate per-application profiles for libraries and per-application profiles for the kernel and kernel modules.

If **--separate=library** is used, the sample file name includes the name of the executable as well as the name of the library.



NOTE

These configuration changes will take effect when the OProfile profiler is restarted.

29.3. STARTING AND STOPPING OPROFILE

To start monitoring the system with OProfile, execute the following command as root:

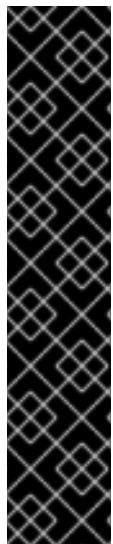
```
~]# opcontrol --start
```

Output similar to the following is displayed:

```
Using log file /var/lib/oprofile/oprofiled.log Daemon started. Profiler running.
```

The settings in **/root/.oprofile/daemonrc** are used.

The OProfile daemon, **oprofiled**, is started; it periodically writes the sample data to the **/var/lib/oprofile/samples/** directory. The log file for the daemon is located at **/var/lib/oprofile/oprofiled.log**.



IMPORTANT

On a Red Hat Enterprise Linux 6 system, the **nmi_watchdog** registers with the **perf** subsystem. Due to this, the **perf** subsystem grabs control of the performance counter registers at boot time, blocking OProfile from working.

To resolve this, either boot with the **nmi_watchdog=0** kernel parameter set, or run the following command to disable **nmi_watchdog** at run time:

```
~]# echo 0 > /proc/sys/kernel/nmi_watchdog
```

To re-enable **nmi_watchdog**, use the following command:

```
~]# echo 1 > /proc/sys/kernel/nmi_watchdog
```

To stop the profiler, execute the following command as root:

```
~]# opcontrol --shutdown
```

29.4. SAVING DATA

Sometimes it is useful to save samples at a specific time. For example, when profiling an executable, it may be useful to gather different samples based on different input data sets. If the number of events to

be monitored exceeds the number of counters available for the processor, multiple runs of OProfile can be used to collect data, saving the sample data to different files each time.

To save the current set of sample files, execute the following command, replacing *name* with a unique descriptive name for the current session.

```
~]# opcontrol --save=name
```

The directory **/var/lib/oprofile/samples/*name*** is created and the current sample files are copied to it.

29.5. ANALYZING THE DATA

Periodically, the OProfile daemon, **oprofiled**, collects the samples and writes them to the **/var/lib/oprofile/samples/** directory. Before reading the data, make sure all data has been written to this directory by executing the following command as root:

```
~]# opcontrol --dump
```

Each sample file name is based on the name of the executable. For example, the samples for the default event on a Pentium III processor for **/bin/bash** becomes:

```
\{root\}/bin/bash\{dep\}\{root\}/bin/bash/CPU_CLK_UNHALTED.100000
```

The following tools are available to profile the sample data once it has been collected:

- **opreport**
- **opannotate**

Use these tools, along with the binaries profiled, to generate reports that can be further analyzed.



WARNING

The executable being profiled must be used with these tools to analyze the data. If it must change after the data is collected, back up the executable used to create the samples as well as the sample files. Please note that the sample file and the binary have to agree. Making a backup is not going to work if they do not match. **oparchive** can be used to address this problem.

Samples for each executable are written to a single sample file. Samples from each dynamically linked library are also written to a single sample file. While OProfile is running, if the executable being monitored changes and a sample file for the executable exists, the existing sample file is automatically deleted. Thus, if the existing sample file is needed, it must be backed up, along with the executable used to create it before replacing the executable with a new version. The OProfile analysis tools use the executable file that created the samples during analysis. If the executable changes the analysis tools will be unable to analyze the associated samples. See [Section 29.4, "Saving Data"](#) for details on how to back up the sample file.

29.5.1. Using oreport

The **oprofile** tool provides an overview of all the executables being profiled.

The following is part of a sample output:

```
Profiling through timer interrupt
TIMER:0|
samples| %|
-----
25926 97.5212 no-vmlinux
359 1.3504 pi
65 0.2445 Xorg
62 0.2332 libvte.so.4.4.0
56 0.2106 libc-2.3.4.so
34 0.1279 libglib-2.0.so.0.400.7
19 0.0715 libXft.so.2.1.2
17 0.0639 bash
8 0.0301 ld-2.3.4.so
8 0.0301 libgdk-x11-2.0.so.0.400.13
6 0.0226 libgobject-2.0.so.0.400.7
5 0.0188 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 **oprofile** man page for a list of available command-line options, such as the **-r** option used to sort the output from the executable with the smallest number of samples to the one with the largest number of samples.

29.5.2. Using oreport on a Single Executable

To retrieve more detailed profiled information about a specific executable, use **oprofile**:

```
~]# oprofile mode executable
```

executable must be the full path to the executable to be analyzed. *mode* must be one of the following:

-l

List sample data by symbols. For example, the following is part of the output from running the command **oprofile -l /lib/tls/libc-version.so**:

```
samples % symbol name
```

```

12 21.4286 __gconv_transform_utf8_internal
5 8.9286 _int_malloc 4 7.1429 malloc
3 5.3571 __i686.get_pc_thunk.bx
3 5.3571 __dl_mcount_wrapper_check
3 5.3571 mbrtowc
3 5.3571 memcpy
2 3.5714 _int_realloc
2 3.5714 __nl_intern_locale_data
2 3.5714 free
2 3.5714 strcmp
1 1.7857 __ctype_get_mb_cur_max
1 1.7857 __unregister_atfork
1 1.7857 __write_nocancel
1 1.7857 __dl_addr
1 1.7857 _int_free
1 1.7857 __itoa_word
1 1.7857 calc_eclosure_iter
1 1.7857 fopen@@@GLIBC_2.1
1 1.7857 getpid
1 1.7857 memmove
1 1.7857 msort_with_tmp
1 1.7857 strcpy
1 1.7857 strlen
1 1.7857 vfprintf
1 1.7857 write

```

The first column is the number of samples for the symbol, the second column is the percentage of samples for this symbol relative to the overall samples for the executable, and the third column is the symbol name.

To sort the output from the largest number of samples to the smallest (reverse order), use **-r** in conjunction with the **-l** option.

-i symbol-name

List sample data specific to a symbol name. For example, the following output is from the command **opreport -l -i __gconv_transform_utf8_internal /lib/tls/libc-version.so**:

```

samples % symbol name
12 100.000 __gconv_transform_utf8_internal

```

The first line is a summary for the symbol/executable combination.

The first column is the number of samples for the memory symbol. The second column is the percentage of samples for the memory address relative to the total number of samples for the symbol. The third column is the symbol name.

-d

List sample data by symbols with more detail than **-l**. For example, the following output is from the command **opreport -l -d __gconv_transform_utf8_internal /lib/tls/libc-version.so**:

```

vma samples % symbol name
00a98640 12 100.000 __gconv_transform_utf8_internal
00a98640 1 8.3333
00a9868c 2 16.6667

```

```
00a9869a 1 8.3333
00a986c1 1 8.3333
00a98720 1 8.3333
00a98749 1 8.3333
00a98753 1 8.3333
00a98789 1 8.3333
00a98864 1 8.3333
00a98869 1 8.3333
00a98b08 1 8.3333
```

The data is the same as the **-l** option except that for each symbol, each virtual memory address used is shown. For each virtual memory address, the number of samples and percentage of samples relative to the number of samples for the symbol is displayed.

-x symbol-name

Exclude the comma-separated list of symbols from the output.

session:name

Specify the full path to the session or a directory relative to the **/var/lib/oprofile/samples/** directory.

29.5.3. Getting more detailed output on the modules

OProfile collects data on a system-wide basis for kernel- and user-space code running on the machine. However, once a module is loaded into the kernel, the information about the origin of the kernel module is lost. The module could have come from the **initrd** file on boot up, the directory with the various kernel modules, or a locally created kernel module. As a result, when OProfile records sample for a module, it just lists the samples for the modules for an executable in the root directory, but this is unlikely to be the place with the actual code for the module. You will need to take some steps to make sure that analysis tools get the executable.

To get a more detailed view of the actions of the module, you will need to either have the module "unstripped" (that is installed from a custom build) or have the **debuginfo** package installed for the kernel.

Find out which kernel is running with the **uname -a** command, obtain the appropriate **debuginfo** package and install it on the machine.

Then proceed with clearing out the samples from previous runs with the following command:

```
~]# opcontrol --reset
```

To start the monitoring process, for example, on a machine with Westmere processor, run the following command:

```
~]# opcontrol --setup --vmlinux=/usr/lib/debug/lib/modules/`uname -r`/vmlinux --
event=CPU_CLK_UNHALTED:500000
```

Then the detailed information, for instance, for the ext4 module can be obtained with:

```
~]# opreport /ext4 -l --image-path /lib/modules/`uname -r`/kernel
CPU: Intel Westmere microarchitecture, speed 2.667e+06 MHz (estimated)
Counted CPU_CLK_UNHALTED events (Clock cycles when not halted) with a unit mask of 0x00 (No
unit mask) count 500000
```

```
warning: could not check that the binary file /lib/modules/2.6.32-191.el6.x86_64/kernel/fs/ext4/ext4.ko
has not been modified since the profile was taken. Results may be inaccurate.

samples %      symbol name
1622  9.8381 ext4_iget
1591  9.6500 ext4_find_entry
1231  7.4665 __ext4_get_inode_loc
783   4.7492 ext4_ext_get_blocks
752   4.5612 ext4_check_dir_entry
644   3.9061 ext4_mark_iloc_dirty
583   3.5361 ext4_get_blocks
583   3.5361 ext4_xattr_get
479   2.9053 ext4_htree_store_dirent
469   2.8447 ext4_get_group_desc
414   2.5111 ext4_dx_find_entry
```

29.5.4. Using **opannotate**

The **opannotate** tool tries to match the samples for particular instructions to the corresponding lines in the source code. The resulting files generated should have the samples for the lines at the left. It also puts in a comment at the beginning of each function listing the total samples for the function.

For this utility to work, the appropriate debuginfo package for the executable must be installed on the system. On Red Hat Enterprise Linux, the debuginfo packages are not automatically installed with the corresponding packages that contain the executable. You have to obtain and install them separately.

The general syntax for **opannotate** is as follows:

```
~]# opannotate --search-dirs src-dir --source executable
```

The directory containing the source code and the executable to be analyzed must be specified. See the **opannotate** man page for a list of additional command-line options.

29.6. UNDERSTANDING /DEV/OPROFILE/

The **/dev/oprofile/** directory contains the file system for OProfile. Use the **cat** command to display the values of the virtual files in this file system. For example, the following command displays the type of processor OProfile detected:

```
~]# cat /dev/oprofile/cpu_type
```

A directory exists in **/dev/oprofile/** for each counter. For example, if there are 2 counters, the directories **/dev/oprofile/0/** and **dev/oprofile/1/** exist.

Each directory for a counter contains the following files:

- **count** – The interval between samples.
- **enabled** – If 0, the counter is off and no samples are collected for it; if 1, the counter is on and samples are being collected for it.
- **event** – The event to monitor.
- **extra** – Used on machines with Nehalem processors to further specify the event to monitor.

- **kernel** – If 0, samples are not collected for this counter event when the processor is in kernel-space; if 1, samples are collected even if the processor is in kernel-space.
- **unit_mask** – Defines which unit masks are enabled for the counter.
- **user** – If 0, samples are not collected for the counter event when the processor is in user-space; if 1, samples are collected even if the processor is in user-space.

The values of these files can be retrieved with the **cat** command. For example:

```
~]# cat /dev/oprofile/0/count
```

29.7. EXAMPLE USAGE

While OProfile can be used by developers to analyze application performance, it can also be used by system administrators to perform system analysis. For example:

- *Determine which applications and services are used the most on a system* – **opreport** can be used to determine how much processor time an application or service uses. If the system is used for multiple services but is under performing, the services consuming the most processor time can be moved to dedicated systems.
- *Determine processor usage* – The **CPU_CLK_UNHALTED** event can be monitored to determine the processor load over a given period of time. This data can then be used to determine if additional processors or a faster processor might improve system performance.

29.8. OPROFILE SUPPORT FOR JAVA

OProfile allows you to profile dynamically compiled code (also known as "just-in-time" or JIT code) of the Java Virtual Machine (JVM). OProfile in Red Hat Enterprise Linux 6 includes built-in support for the JVM Tools Interface (JVMTI) agent library, which supports Java 1.5 and higher.

29.8.1. Profiling Java Code

To profile JIT code from the Java Virtual Machine with the JVMTI agent, add the following to the JVM startup parameters:

```
-agentlib:jvmti_oprofile
```



NOTE

The oprofile-jit package must be installed on the system in order to profile JIT code with OProfile.

To learn more about Java support in OProfile, see the OProfile Manual, which is linked from [Section 29.11, "Additional Resources"](#).

29.9. GRAPHICAL INTERFACE

Some OProfile preferences can be set with a graphical interface. To start it, execute the **oprof_start** command as root at a shell prompt. To use the graphical interface, you will need to have the oprofile-gui package installed.

After changing any of the options, save them by clicking the **Save and quit** button. The preferences are written to **/root/.oprofile/daemonrc**, and the application exits.



NOTE

Exiting the application does not stop OProfile from sampling.

On the **Setup** tab, to set events for the processor counters as discussed in [Section 29.2.2, "Setting Events to Monitor"](#), select the counter from the pulldown menu and select the event from the list. A brief description of the event appears in the text box below the list. Only events available for the specific counter and the specific architecture are displayed. The interface also displays whether the profiler is running and some brief statistics about it.

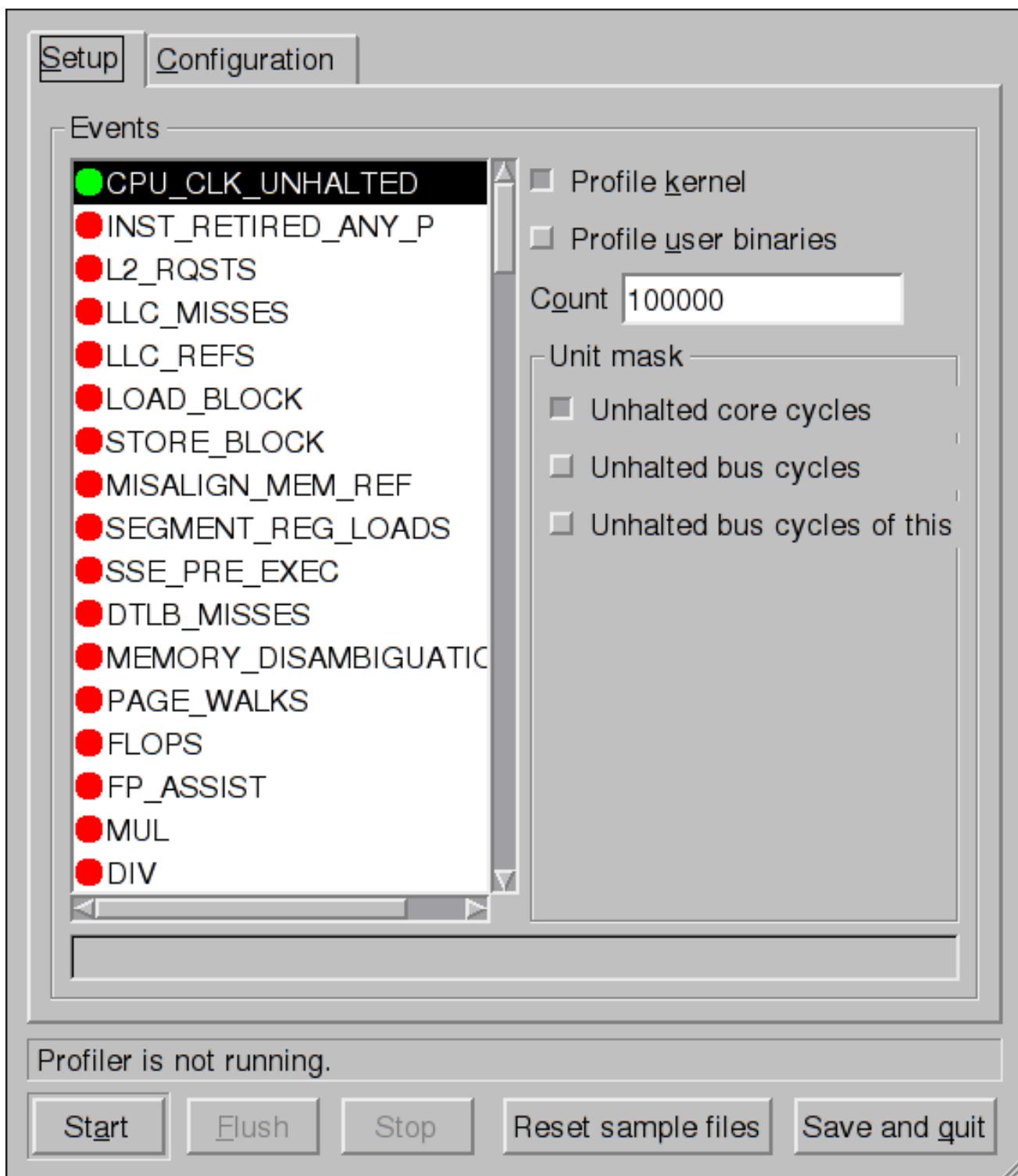


Figure 29.1. OProfile Setup

On the right side of the tab, select the **Profile kernel** option to count events in kernel mode for the currently selected event, as discussed in [Section 29.2.3, "Separating Kernel and User-space Profiles"](#). If this option is unselected, no samples are collected for the kernel.

Select the **Profile user binaries** option to count events in user mode for the currently selected event, as discussed in [Section 29.2.3, "Separating Kernel and User-space Profiles"](#). If this option is unselected, no samples are collected for user applications.

Use the **Count** text field to set the sampling rate for the currently selected event as discussed in [Section 29.2.2.1, "Sampling Rate"](#).

If any unit masks are available for the currently selected event, as discussed in [Section 29.2.2.2, "Unit Masks"](#), they are displayed in the **Unit Masks** area on the right side of the **Setup** tab. Select the check box beside the unit mask to enable it for the event.

On the **Configuration** tab, to profile the kernel, enter the name and location of the **vmlinux** file for the kernel to monitor in the **Kernel image file** text field. To configure OProfile not to monitor the kernel, select **No kernel image**.

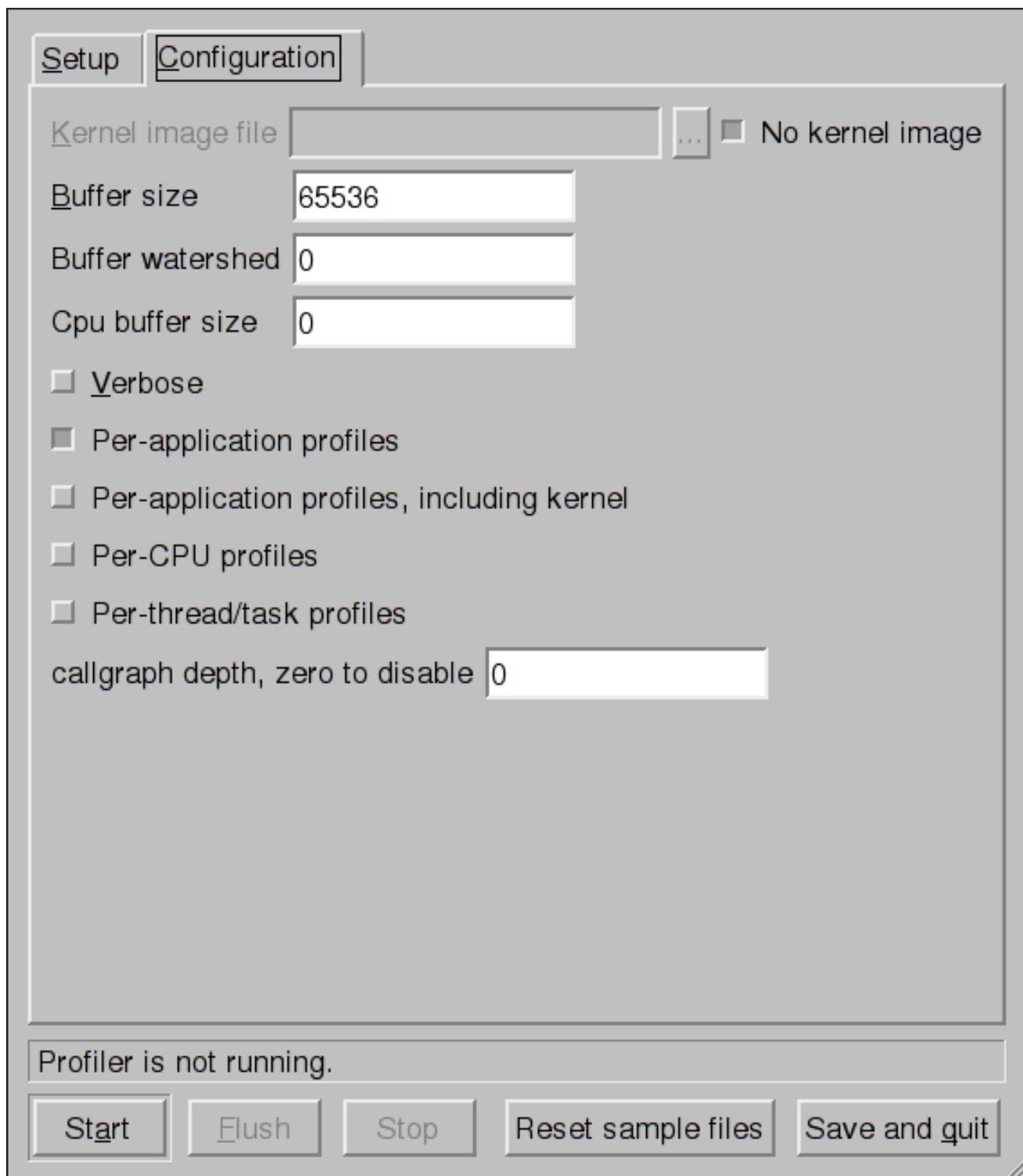


Figure 29.2. OProfile Configuration

If the **Verbose** option is selected, the **oprofiled** daemon log includes more information.

If **Per-application profiles** is selected, OProfile generates per-application profiles for libraries. This is

equivalent to the **opcontrol --separate=library** command. If **Per-application profiles, including kernel** is selected, OProfile generates per-application profiles for the kernel and kernel modules as discussed in [Section 29.2.3, “Separating Kernel and User-space Profiles”](#). This is equivalent to the **opcontrol --separate=kernel** command.

To force data to be written to samples files as discussed in [Section 29.5, “Analyzing the Data”](#), click the **Flush** button. This is equivalent to the **opcontrol --dump** command.

To start OProfile from the graphical interface, click **Start**. To stop the profiler, click **Stop**. Exiting the application does not stop OProfile from sampling.

29.10. OPROFILE AND SYSTEMTAP

SystemTap is a tracing and probing tool that allows users to study and monitor the activities of the operating system in fine detail. It provides information similar to the output of tools like **netstat**, **ps**, **top**, and **iostat**; however, SystemTap is designed to provide more filtering and analysis options for collected information.

While using OProfile is suggested in cases of collecting data on where and why the processor spends time in a particular area of code, it is less usable when finding out why the processor stays idle.

You might want to use SystemTap when instrumenting specific places in code. Because SystemTap allows you to run the code instrumentation without having to stop and restart the instrumentation, it is particularly useful for instrumenting the kernel and daemons.

For more information on SystemTap, see [Section 29.11.2, “Useful Websites”](#) for the relevant SystemTap documentation.

29.11. ADDITIONAL RESOURCES

This chapter only highlights OProfile and how to configure and use it. To learn more, see the following resources.

29.11.1. Installed Docs

- **/usr/share/doc/oprofile-version/oprofile.html** – *OProfile Manual*
- **oprofile** man page – Discusses **opcontrol**, **opreport**, **opannotate**, and **ophelp**

29.11.2. Useful Websites

- <http://oprofile.sourceforge.net/> – Contains the latest documentation, mailing lists, IRC channels, and more.
- [SystemTap Beginners Guide](#) – Provides basic instructions on how to use SystemTap to monitor different subsystems of Red Hat Enterprise Linux in finer detail.

PART VIII. KERNEL, MODULE AND DRIVER CONFIGURATION

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

CHAPTER 30. MANUALLY UPGRADING THE KERNEL

The Red Hat Enterprise Linux kernel is custom-built by the Red Hat Enterprise Linux kernel team to ensure its integrity and compatibility with supported hardware. Before Red Hat releases a kernel, it must first pass a rigorous set of quality assurance tests.

Red Hat Enterprise Linux kernels are packaged in the RPM format so that they are easy to upgrade and verify using the **Yum** or **PackageKit** package managers. **PackageKit** automatically queries the Red Hat Network servers and informs you of packages with available updates, including kernel packages.

This chapter is therefore *only* useful for users who need to manually update a kernel package using the **rpm** command instead of **yum**.



WARNING

Whenever possible, use either the **Yum** or **PackageKit** package manager to install a new kernel because they always *install* a new kernel instead of replacing the current one, which could potentially leave your system unable to boot.



WARNING

Building a custom kernel is not supported by the Red Hat Global Services Support team, and therefore is not explored in this manual.

For more information on installing kernel packages with **Yum**, see [Section 8.1.2, “Updating Packages”](#). For information on Red Hat Network, see [Chapter 6, Registering the System and Managing Subscriptions](#).

30.1. OVERVIEW OF KERNEL PACKAGES

Red Hat Enterprise Linux contains the following kernel packages:

- **kernel** – Contains the kernel for single, multicore and multiprocessor systems.
- **kernel-debug** – Contains a kernel with numerous debugging options enabled for kernel diagnosis, at the expense of reduced performance.
- **kernel-devel** – Contains the kernel headers and makefiles sufficient to build modules against the kernel package.
- **kernel-debug-devel** – Contains the development version of the kernel with numerous debugging options enabled for kernel diagnosis, at the expense of reduced performance.

- kernel-doc – Documentation files from the kernel source. Various portions of the Linux kernel and the device drivers shipped with it are documented in these files. Installation of this package provides a reference to the options that can be passed to Linux kernel modules at load time.

By default, these files are placed in the `/usr/share/doc/kernel-doc-<kernel_version>` directory.

- kernel-headers – Includes the C header files that specify the interface between the Linux kernel and user-space libraries and programs. The header files define structures and constants that are needed for building most standard programs.
- kernel-firmware – Contains all of the firmware files that are required by various devices to operate.
- perf – This package contains supporting scripts and documentation for the `perf` tool shipped in each kernel image subpackage.

30.2. PREPARING TO UPGRADE

Before upgrading the kernel, it is recommended that you take some precautionary steps.

First, ensure that working boot media exists for the system. If the boot loader is not configured properly to boot the new kernel, you can use this media to boot into Red Hat Enterprise Linux.

USB media often comes in the form of flash devices sometimes called *pen drives*, *thumb disks*, or *keys*, or as an externally-connected hard disk device. Almost all media of this type is formatted as a **VFAT** file system. You can create bootable USB media on media formatted as **ext2**, **ext3**, or **VFAT**.

You can transfer a distribution image file or a minimal boot media image file to USB media. Make sure that sufficient free space is available on the device. Around **4 GB** is required for a distribution DVD image, around **700 MB** for a distribution CD image, or around **10 MB** for a minimal boot media image.

You must have a copy of the **boot.iso** file from a Red Hat Enterprise Linux installation DVD, or installation CD-ROM #1, and you need a USB storage device formatted with the **VFAT** file system and around **16 MB** of free space. The following procedure will not affect existing files on the USB storage device unless they have the same path names as the files that you copy onto it. To create USB boot media, perform the following commands as **root**:

1. Install the **SYSLINUX** boot loader on the USB storage device:

```
~]# syslinux /dev/sdX1
```

...where `sdX` is the device name.

2. Create mount points for **boot.iso** and the USB storage device:

```
~]# mkdir /mnt/isoboot /mnt/diskboot
```

3. Mount **boot.iso**:

```
~]# mount -o loop boot.iso /mnt/isoboot
```

4. Mount the USB storage device:

```
~]# mount /dev/<sdX1> /mnt/diskboot
```

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 **yum list installed "kernel-**"** at a shell prompt. The output will comprise some or all of the following packages, depending on the system's architecture, and the version numbers may differ:

```
~]# yum list installed "kernel-**"
kernel.x86_64          2.6.32-17.el6      @rhel-x86_64-server-6
kernel-doc.noarch        2.6.32-17.el6      @rhel-x86_64-server-6
kernel-firmware.noarch   2.6.32-17.el6      @rhel-x86_64-server-6
kernel-headers.x86_64    2.6.32-17.el6      @rhel-x86_64-server-6
```

From the output, determine which packages need to be downloaded for the kernel upgrade. For a single processor system, the only required package is the kernel package. See [Section 30.1, “Overview of Kernel Packages”](#) for descriptions of the different packages.

30.3. DOWNLOADING THE UPGRADED KERNEL

There are several ways to determine if an updated kernel is available for the system.

- Security Errata – See <http://www.redhat.com/security/updates/> for information on security errata, including kernel upgrades that fix security issues.
- The Red Hat Network – For a system subscribed to the Red Hat Network, the **yum** package manager can download the latest kernel and upgrade the kernel on the system. The **Dracut** utility will create an initial RAM disk image if needed, and configure the boot loader to boot the new kernel. For more information on installing packages from the Red Hat Network, see [Chapter 8, Yum](#). For more information on subscribing a system to the Red Hat Network, see [Chapter 6, Registering the System and Managing Subscriptions](#).

If **yum** was used to download and install the updated kernel from the Red Hat Network, follow the instructions in [Section 30.5, “Verifying the Initial RAM Disk Image”](#) and [Section 30.6, “Verifying the Boot Loader”](#), only do not change the kernel to boot by default. Red Hat Network automatically changes the default kernel to the latest version. To install the kernel manually, continue to [Section 30.4, “Performing the Upgrade”](#).

30.4. PERFORMING THE UPGRADE

After retrieving all of the necessary packages, it is time to upgrade the existing kernel.



IMPORTANT

It is strongly recommended that you keep the old kernel in case there are problems with the new kernel.

At a shell prompt, change to the directory that contains the kernel RPM packages. Use **-i** argument with the **rpm** command to keep the old kernel. Do *not* use the **-U** option, since it overwrites the currently installed kernel, which creates boot loader problems. For example:

```
~]# rpm -ivh kernel-<kernel_version>.<arch>.rpm
```

The next step is to verify that the initial RAM disk image has been created. See [Section 30.5, “Verifying the Initial RAM Disk Image”](#) for details.

30.5. VERIFYING THE INITIAL RAM DISK IMAGE

The job of the initial RAM disk image is to preload the block device modules, such as for IDE, SCSI or RAID, so that the root file system, on which those modules normally reside, can then be accessed and mounted. On Red Hat Enterprise Linux 6 systems, whenever a new kernel is installed using either the **Yum**, **PackageKit**, or **RPM** package manager, the **Dracut** utility is always called by the installation scripts to create an *initramfs* (initial RAM disk image).

On all architectures other than IBM eServer System i (see [the section called “Verifying the Initial RAM Disk Image and Kernel on IBM eServer System i”](#)), you can create an **initramfs** by running the **dracut** command. However, you usually don’t need to create an **initramfs** manually: this step is automatically performed if the kernel and its associated packages are installed or upgraded from RPM packages distributed by Red Hat.

You can verify that an **initramfs** corresponding to your current kernel version exists and is specified correctly in the **grub.conf** configuration file by following this procedure:

Procedure 30.1. Verifying the Initial RAM Disk Image

- 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 30.1. Ensuring that the kernel and initramfs versions match

```
~]# ls /boot/
config-2.6.32-17.el6.x86_64      lost+found
config-2.6.32-19.el6.x86_64      symvers-2.6.32-17.el6.x86_64.gz
config-2.6.32-22.el6.x86_64      symvers-2.6.32-19.el6.x86_64.gz
efi                           symvers-2.6.32-22.el6.x86_64.gz
grub                         System.map-2.6.32-17.el6.x86_64
initramfs-2.6.32-17.el6.x86_64.img  System.map-2.6.32-19.el6.x86_64
initramfs-2.6.32-19.el6.x86_64.img  System.map-2.6.32-22.el6.x86_64
initramfs-2.6.32-22.el6.x86_64.img  vmlinuz-2.6.32-17.el6.x86_64
```

```
initrd-2.6.32-17.el6.x86_64kdump.img vmlinuz-2.6.32-19.el6.x86_64
initrd-2.6.32-19.el6.x86_64kdump.img vmlinuz-2.6.32-22.el6.x86_64
initrd-2.6.32-22.el6.x86_64kdump.img
```

Example 30.1, “Ensuring that the kernel and initramfs versions match” shows that:

- we have three kernels installed (or, more correctly, three kernel files are present in **/boot/**),
- the latest kernel is **vmlinuz-2.6.32-22.el6.x86_64**, and
- an **initramfs** file matching our kernel version, **initramfs-2.6.32-22.el6.x86_64.img**, also exists.



IMPORTANT

In the **/boot/** directory you may find several **initrd-<version>kdump.img** files. These are special files created by the **Kdump** mechanism for kernel debugging purposes, are not used to boot the system, and can safely be ignored.

2. (Optional) If your **initramfs-<kernel_version>** file does not match the version of the latest kernel in **/boot/**, or, in certain other situations, you may need to generate an **initramfs** file with the **Dracut** utility. Simply invoking **dracut** as root without options causes it to generate an **initramfs** file in the **/boot/** directory for the latest kernel present in that directory:

```
~]# dracut
```

You must use the **--force** option if you want **dracut** to overwrite an existing **initramfs** (for example, if your **initramfs** has become corrupt). Otherwise **dracut** will refuse to overwrite the existing **initramfs** file:

```
~]# dracut
```

Will not override existing initramfs (/boot/initramfs-2.6.32-22.el6.x86_64.img) without --force

You can create an initramfs in the current directory by calling **dracut <initramfs_name> <kernel_version>**:

```
~]# dracut "initramfs-$(uname -r).img" $(uname -r)
```

If you need to specify specific kernel modules to be preloaded, add the names of those modules (minus any file name suffixes such as **.ko**) inside the parentheses of the **add_dracutmodules="<module> [<more_modules>]"** directive of the **/etc/dracut.conf** configuration file. You can list the file contents of an **initramfs** image file created by dracut by using the **lsinitrd <initramfs_file>** command:

```
~]# lsinitrd initramfs-2.6.32-22.el6.x86_64.img
initramfs-2.6.32-22.el6.x86_64.img:
=====
```

```
dracut-004-17.el6
=====
```

```
drwxr-xr-x 23 root root 0 May 3 22:34 .
```

```

drwxr-xr-x 2 root  root      0 May  3 22:33 proc
-rwxr-xr-x 1 root  root    7575 Mar 25 19:53 init
drwxr-xr-x 7 root  root      0 May  3 22:34 etc
drwxr-xr-x 2 root  root      0 May  3 22:34 etc/modprobe.d
[output truncated]

```

See **man dracut** and **man dracut.conf** for more information on options and usage.

3. Examine the **grub.conf** configuration file in the **/boot/grub/** directory to ensure that an **initrd initramfs-<kernel_version>.img** exists for the kernel version you are booting. See [Section 30.6, "Verifying the Boot Loader"](#) for more information.

Verifying the Initial RAM Disk Image and Kernel on IBM eServer System i

On IBM eServer System i machines, the initial RAM disk and kernel files are combined into a single file, which is created with the **addRamDisk** command. This step is performed automatically if the kernel and its associated packages are installed or upgraded from the RPM packages distributed by Red Hat; thus, it does not need to be executed manually. To verify that it was created, use the command **ls -l /boot/** to make sure the **/boot/vmlinuinitrd-<kernel_version>** file already exists (the **<kernel_version>** should match the version of the kernel just installed).

30.6. VERIFYING THE BOOT LOADER

When you install a kernel using **rpm**, the kernel package creates an entry in the boot loader configuration file for that new kernel. However, **rpm** does *not* configure the new kernel to boot as the default kernel. You must do this manually when installing a new kernel with **rpm**.

It is always recommended to double-check the boot loader configuration file after installing a new kernel with **rpm** to ensure that the configuration is correct. Otherwise, the system may not be able to boot into Red Hat Enterprise Linux properly. If this happens, boot the system with the boot media created earlier and re-configure the boot loader.

In the following table, find your system's architecture to determine the boot loader it uses, and then click on the "See" link to jump to the correct instructions for your system.

Table 30.1. Boot loaders by architecture

Architecture	Boot Loader	See
x86	GRUB	Section 30.6.1, "Configuring the GRUB Boot Loader"
AMD AMD64 or Intel 64	GRUB	Section 30.6.1, "Configuring the GRUB Boot Loader"
IBM eServer System i	OS/400	Section 30.6.3, "Configuring the OS/400 Boot Loader"
IBM eServer System p	YABOOT	Section 30.6.4, "Configuring the YABOOT Boot Loader"
IBM System z	z/IPL	

30.6.1. Configuring the GRUB Boot Loader

GRUB's configuration file, **/boot/grub/grub.conf**, contains a few lines with directives, such as **default**, **timeout**, **splashimage** and **hiddenmenu** (the last directive has no argument). The remainder of the file contains 4-line *stanzas* that each refer to an installed kernel. These stanzas always start with a **title** entry, after which the associated **root**, **kernel** and **initrd** directives should always be indented. Ensure that each stanza starts with a **title** that contains a version number (in parentheses) that matches the version number in the **kernel /vmlinuz-<version_number>** line of the same stanza.

Example 30.2. /boot/grub/grub.conf

```
# grub.conf generated by anaconda
[comments omitted]
default=1
timeout=0
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu

title Red Hat Enterprise Linux (2.6.32-22.el6.x86_64)
    root (hd0,0)
    kernel /vmlinuz-2.6.32-22.el6.x86_64 ro root=/dev/mapper/vg_vm6b-lv_root
    rd_LVM_LV=vg_vm6b/lv_root rd_NO_LUKS rd_NO_MD rd_NO_DM LANG=en_US.UTF-8
    SYSFONT=latarcyrheb-sun16 KEYBOARDTYPE=pc KEYTABLE=us rhgb quiet crashkernel=auto
    initrd /initramfs-2.6.32-22.el6.x86_64.img

title Red Hat Enterprise Linux (2.6.32-19.el6.x86_64)
    root (hd0,0)
    kernel /vmlinuz-2.6.32-19.el6.x86_64 ro root=/dev/mapper/vg_vm6b-lv_root
    rd_LVM_LV=vg_vm6b/lv_root rd_NO_LUKS rd_NO_MD rd_NO_DM LANG=en_US.UTF-8
    SYSFONT=latarcyrheb-sun16 KEYBOARDTYPE=pc KEYTABLE=us rhgb quiet crashkernel=auto
    initrd /initramfs-2.6.32-19.el6.x86_64.img

title Red Hat Enterprise Linux 6 (2.6.32-17.el6.x86_64)
    root (hd0,0)
    kernel /vmlinuz-2.6.32-17.el6.x86_64 ro root=/dev/mapper/vg_vm6b-lv_root
    rd_LVM_LV=vg_vm6b/lv_root rd_NO_LUKS rd_NO_MD rd_NO_DM LANG=en_US.UTF-8
    SYSFONT=latarcyrheb-sun16 KEYBOARDTYPE=pc KEYTABLE=us rhgb quiet
    initrd /initramfs-2.6.32-17.el6.x86_64.img
```

If a separate **/boot/** partition was created, the paths to the kernel and the **initramfs** image are relative to **/boot/**. This is the case in [Example 30.2, "/boot/grub/grub.conf"](#), above. Therefore the **initrd /initramfs-2.6.32-22.el6.x86_64.img** line in the first kernel stanza means that the **initramfs** image is actually located at **/boot/initramfs-2.6.32-22.el6.x86_64.img** when the root file system is mounted, and likewise for the kernel path (for example: **kernel /vmlinuz-2.6.32-22.el6.x86_64**) in each stanza of **grub.conf**.



NOTE

In kernel boot stanzas in **grub.conf**, the **initrd** directive must point to the location (relative to the **/boot** directory if it is on a separate partition) of the **initramfs** file corresponding to the same kernel version. This directive is called **initrd** because the previous tool which created initial RAM disk images, **mkintrd**, created what were known as **initrd** files. Thus the **grub.conf** directive remains **initrd** to maintain compatibility with other tools. The file-naming convention of systems using the **dracut** utility to create the initial RAM disk image is: **initramfs-<kernel_version>.img**

Dracut is a new utility available in Red Hat Enterprise Linux 6, and much-improved over **mkintrd**. For information on using **Dracut**, see [Section 30.5, “Verifying the Initial RAM Disk Image”](#).

You should ensure that the kernel version number as given on the **kernel /vmlinuz-<kernel_version>** line matches the version number of the **initramfs** image given on the **initrd /initramfs-<kernel_version>.img** line of each stanza. See [Procedure 30.1, “Verifying the Initial RAM Disk Image”](#) for more information.

The **default=** directive tells GRUB which kernel to boot *by default*. Each **title** in **grub.conf** represents a bootable kernel. GRUB counts the **title**d stanzas representing bootable kernels starting with **0**. In [Example 30.2, “/boot/grub/grub.conf”](#), the line **default=1** indicates that GRUB will boot, by default, the second kernel entry, i.e. **title Red Hat Enterprise Linux (2.6.32-19.el6.x86_64)**.

In [Example 30.2, “/boot/grub/grub.conf”](#) GRUB is therefore configured to boot an older kernel, when we compare by version numbers. In order to boot the newer kernel, which is the *first title* entry in **grub.conf**, we would need to change the **default** value to **0**.

After installing a new kernel with **rpm**, verify that **/boot/grub/grub.conf** is correct, change the **default=** value to the new kernel (while remembering to count from **0**), and reboot the computer into the new kernel. Ensure your hardware is detected by watching the boot process output.

If GRUB presents an error and is unable to boot into the default kernel, it is often easiest to try to boot into an alternative or older kernel so that you can fix the problem.



IMPORTANT

If you set the **timeout** directive in **grub.conf** to **0**, GRUB will not display its list of bootable kernels when the system starts up. In order to display this list when booting, press and hold any alphanumeric key while and immediately after BIOS information is displayed. GRUB will present you with the GRUB menu.

Alternatively, use the boot media you created earlier to boot the system.

30.6.2. Configuring the Loopback Device Limit

The maximum number of loopback devices in Red Hat Enterprise Linux 6 is set by the **max_loop** kernel option. For example, to set the maximum number of loopback devices to **64**, edit the **/etc/grub.conf** file, and add **max_loop=64** at the end of the kernel line. The line in **/etc/grub.conf** would then look something like this:

```
kernel /vmlinuz-2.6.32-131.0.15.el6.x86_64 ro root=/dev/mapper/root rhgb quiet max_loop=64
initrd /initramfs-2.6.32-131.0.15.el6.x86_64.img
```

Reboot the system for the changes to take affect.

By default, eight **/dev/loop*** devices (**/dev/loop0** to **/dev/loop7**) are automatically generated, but others can be created as desired. For example, to set up a ninth loop device named **/dev/loop8**, issue the following command as **root**:

```
~]# mknod /dev/loop8 b 7 8
```

Thus, an administrator on a system with a Red Hat Enterprise Linux 6 kernel can create the desired number of loopback devices manually, with an init script, or with a **udev** rule.

However, if **max_loop** has been set before the system booted, **max_loop** becomes a hard limit on the number of loopback devices, and the number of loopback devices cannot be dynamically grown beyond the limit.

30.6.3. Configuring the OS/400 Boot Loader

The **/boot/vmlinird-*<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/vmlinird-<kernel-version> of=/proc/iSeries/mf/<side>/vmlinu bs=8k
```

Begin testing the new kernel by rebooting the computer and watching the messages to ensure that the hardware is detected properly.

30.6.4. Configuring the YABOOT Boot Loader

IBM eServer System p uses YABOOT as its boot loader. YABOOT uses **/etc/yaboot.conf** as its configuration file. Confirm that the file contains an **image** section with the same version as the kernel package just installed, and likewise for the **initramfs** image:

```
boot=/dev/sda1 init-message=Welcome to Red Hat Enterprise Linux! Hit <TAB> for boot options
partition=2 timeout=30 install=/usr/lib/yaboot/yaboot delay=10 nonvram
image=/vmlinuz-2.6.32-17.EL
label=old
read-only
initrd=/initramfs-2.6.32-17.EL.img
append="root=LABEL=/"
image=/vmlinuz-2.6.32-19.EL
label=linux
read-only
initrd=/initramfs-2.6.32-19.EL.img
append="root=LABEL=/"
```

Notice that the default is not set to the new kernel. The kernel in the first image is booted by default. To change the default kernel to boot either move its image stanza so that it is the first one listed or add the directive **default** and set it to the **label** of the image stanza that contains the new kernel.

Begin testing the new kernel by rebooting the computer and watching the messages to ensure that the hardware is detected properly.

CHAPTER 31. WORKING WITH KERNEL MODULES

The Linux kernel is modular, which means it can extend its capabilities through the use of dynamically-loaded *kernel modules*. A kernel module can provide:

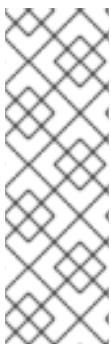
- a device driver which adds support for new hardware; or,
- support for a file system such as **btrfs** or **NFS**.

Like the kernel itself, modules can take parameters that customize their behavior, though the default parameters work well in most cases. User-space tools can list the modules currently loaded into a running kernel; query all available modules for available parameters and module-specific information; and load or unload (remove) modules dynamically into or from a running kernel. Many of these utilities, which are provided by the module-init-tools package, take module dependencies into account when performing operations so that manual dependency-tracking is rarely necessary.

On modern systems, kernel modules are automatically loaded by various mechanisms when the conditions call for it. However, there are occasions when it is necessary to load and/or unload modules manually, such as when a module provides optional functionality, one module should be preferred over another although either could provide basic functionality, or when a module is misbehaving, among other situations.

This chapter explains how to:

- use the user-space module-init-tools package to display, query, load and unload kernel modules and their dependencies;
- set module parameters both dynamically on the command line and permanently so that you can customize the behavior of your kernel modules; and,
- load modules at boot time.



NOTE

In order to use the kernel module utilities described in this chapter, first ensure the module-init-tools package is installed on your system by running, as root:

```
~]# yum install module-init-tools
```

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

31.1. LISTING CURRENTLY-LOADED MODULES

You can list all kernel modules that are currently loaded into the kernel by running the **lsmod** command:

```
~]$ lsmod
Module           Size Used by
xfs              803635  1
exportfs         3424   1 xfs
vfat             8216   1
fat              43410  1 vfat
tun              13014  2
fuse             54749  2
```

```

ip6table_filter      2743 0
ip6_tables          16558 1 ip6table_filter
ebtable_nat         1895 0
ebtables            15186 1 ebtable_nat
ipt_MASQUERADE     2208 6
iptable_nat        5420 1
nf_nat              19059 2 ipt_MASQUERADE,iptable_nat
rfcomm              65122 4
ipv6                267017 33
sco                 16204 2
bridge              45753 0
stp                 1887 1 bridge
llc                 4557 2 bridge,stp
bnep                15121 2
l2cap               45185 16 rfcomm,bnep
cpufreq_ondemand   8420 2
acpi_cpufreq        7493 1
freq_table          3851 2 cpufreq_ondemand,acpi_cpufreq
usb_storage         44536 1
sha256_generic     10023 2
aes_x86_64          7654 5
aes_generic         27012 1 aes_x86_64
cbc                 2793 1
dm_crypt            10930 1
kvm_intel           40311 0
kvm                 253162 1 kvm_intel
[output truncated]

```

Each row of **lsmod** output specifies:

- the name of a kernel module currently loaded in memory;
- the amount of memory it uses; and,
- the sum total of processes that are using the module and other modules which depend on it, followed by a list of the names of those modules, if there are any. Using this list, you can first unload all the modules depending the module you want to unload. For more information, see [Section 31.4, “Unloading a Module”](#).

Finally, note that **lsmod** output is less verbose and considerably easier to read than the content of the **/proc/modules** pseudo-file.

31.2. DISPLAYING INFORMATION ABOUT A MODULE

You can display detailed information about a kernel module by running the **modinfo <module_name>** command.



NOTE

When entering the name of a kernel module as an argument to one of the module-init-tools utilities, do not append a **.ko** extension to the end of the name. Kernel module names do not have extensions: their corresponding files do.

For example, to display information about the **e1000e** module, which is the Intel PRO/1000 network driver, run:

Example 31.1. Listing information about a kernel module with lsmod

```
~]# modinfo e1000e
filename:      /lib/modules/2.6.32-71.el6.x86_64/kernel/drivers/net/e1000e/e1000e.ko
version:       1.2.7-k2
license:        GPL
description:   Intel(R) PRO/1000 Network Driver
author:        Intel Corporation, <linux.nics@intel.com>
srcversion:    93CB73D3995B501872B2982
alias:         pci:v00008086d00001503sv*sd*bc*sc*i*
alias:         pci:v00008086d00001502sv*sd*bc*sc*i*
[some alias lines omitted]
alias:         pci:v00008086d0000105Esv*sd*bc*sc*i*
depends:
vermagic:     2.6.32-71.el6.x86_64 SMP mod_unload modversions
parm:          copybreak:Maximum size of packet that is copied to a new buffer on receive (uint)
parm:          TxIntDelay:Transmit Interrupt Delay (array of int)
parm:          TxAbsIntDelay:Transmit Absolute Interrupt Delay (array of int)
parm:          RxIntDelay:Receive Interrupt Delay (array of int)
parm:          RxAbsIntDelay:Receive Absolute Interrupt Delay (array of int)
parm:          InterruptThrottleRate:Interrupt Throttling Rate (array of int)
parm:          IntMode:Interrupt Mode (array of int)
parm:          SmartPowerDownEnable:Enable PHY smart power down (array of int)
parm:          KumeranLockLoss:Enable Kumeran lock loss workaround (array of int)
parm:          WriteProtectNVM:Write-protect NVM [WARNING: disabling this can lead to corrupted
NVM] (array of int)
parm:          CrcStripping:Enable CRC Stripping, disable if your BMC needs the CRC (array of int)
parm:          EEE:Enable/disable on parts that support the feature (array of int)
```

Here are descriptions of a few of the fields in **modinfo** output:

filename

The absolute path to the **.ko** kernel object file. You can use **modinfo -n** as a shortcut command for printing only the **filename** field.

description

A short description of the module. You can use **modinfo -d** as a shortcut command for printing only the description field.

alias

The **alias** field appears as many times as there are aliases for a module, or is omitted entirely if there are none.

depends

This field contains a comma-separated list of all the modules this module depends on.

**NOTE**

If a module has no dependencies, the **depends** field may be omitted from the output.

parm

Each **parm** field presents one module parameter in the form ***parameter_name:description***, where:

- *parameter_name* is the exact syntax you should use when using it as a module parameter on the command line, or in an option line in a **.conf** file in the **/etc/modprobe.d** directory; and,
- *description* is a brief explanation of what the parameter does, along with an expectation for the type of value the parameter accepts (such as int, unit or array of int) in parentheses.

You can list all parameters that the module supports by using the **-p** option. However, because useful value type information is omitted from **modinfo -p** output, it is more useful to run:

Example 31.2. Listing module parameters

```
~]# modinfo e1000e | grep "parm" | sort
parm:      copybreak:Maximum size of packet that is copied to a new buffer on receive (uint)
parm:      CrcStripping:Enable CRC Stripping, disable if your BMC needs the CRC (array of
int)
parm:      EEE:Enable/disable on parts that support the feature (array of int)
parm:      InterruptThrottleRate:Interrupt Throttling Rate (array of int)
parm:      IntMode:Interrupt Mode (array of int)
parm:      KumeranLockLoss:Enable Kumeran lock loss workaround (array of int)
parm:      RxAbsIntDelay:Receive Absolute Interrupt Delay (array of int)
parm:      RxIntDelay:Receive Interrupt Delay (array of int)
parm:      SmartPowerDownEnable:Enable PHY smart power down (array of int)
parm:      TxAbsIntDelay:Transmit Absolute Interrupt Delay (array of int)
parm:      TxIntDelay:Transmit Interrupt Delay (array of int)
parm:      WriteProtectNVM:Write-protect NVM [WARNING: disabling this can lead to
corrupted NVM] (array of int)
```

31.3. LOADING A MODULE

To load a kernel module, run the **modprobe <module_name>** command as root. For example, to load the **wacom** module, run:

```
~]# modprobe wacom
```

By default, **modprobe** attempts to load the module from the **/lib/modules/<kernel_version>/kernel/drivers/** directory. In this directory, each type of module has its own subdirectory, such as **net/** and **scsi/**, for network and SCSI interface drivers respectively.

Some modules have dependencies, which are other kernel modules that must be loaded before the module in question can be loaded. A list of module dependencies is generated and maintained by the **depmod** program that is run automatically whenever a kernel or driver package is installed on the system. The **depmod** program keeps the list of dependencies in the **/lib/modules/<kernel_version>/modules.dep** file. The **modprobe** command always reads the **modules.dep** file when performing operations. When you ask **modprobe** to load a specific kernel module, it first examines the dependencies of that module, if there are any, and loads them if they are not already loaded into the kernel. **modprobe** resolves dependencies recursively: If necessary, it loads all dependencies of dependencies, and so on, thus ensuring that all dependencies are always met.

You can use the **-v** (or **--verbose**) option to cause **modprobe** to display detailed information about what it is doing, which may include loading module dependencies. The following is an example of loading the **Fibre Channel over Ethernet** module verbosely:

Example 31.3. modprobe -v shows module dependencies as they are loaded

```
~]# modprobe -v fcoe
insmod /lib/modules/2.6.32-71.el6.x86_64/kernel/drivers/scsi/scsi_tgt.ko
insmod /lib/modules/2.6.32-71.el6.x86_64/kernel/drivers/scsi/scsi_transport_fc.ko
insmod /lib/modules/2.6.32-71.el6.x86_64/kernel/drivers/scsi/libfc/libfc.ko
insmod /lib/modules/2.6.32-71.el6.x86_64/kernel/drivers/scsi/fcoe/libfcoe.ko
insmod /lib/modules/2.6.32-71.el6.x86_64/kernel/drivers/scsi/fcoe.ko
```

This example shows that **modprobe** loaded the **scsi_tgt**, **scsi_transport_fc**, **libfc** and **libfcoe** modules as dependencies before finally loading **fcoe**. Also note that **modprobe** used the more “primitive” **insmod** command to insert the modules into the running kernel.



IMPORTANT

Although the **insmod** command can also be used to load kernel modules, it does not resolve dependencies. Because of this, you should *always* load modules using **modprobe** instead.

31.4. UNLOADING A MODULE

You can unload a kernel module by running **modprobe -r <module_name>** as root. For example, assuming that the **wacom** module is already loaded into the kernel, you can unload it by running:

```
~]# modprobe -r wacom
```

However, this command will fail if a process is using:

- the **wacom** module,
- a module that **wacom** directly depends on, or,
- any module that **wacom**—through the dependency tree—depends on indirectly.

See [Section 31.1, “Listing Currently-Loaded Modules”](#) for more information about using **lsmod** to obtain the names of the modules which are preventing you from unloading a certain module.

For example, if you want to unload the **firewire_ohci** module (because you believe there is a bug in it that is affecting system stability, for example), your terminal session might look similar to this:

```
~]# modinfo -F depends firewire_ohci
depends:    firewire-core
~]# modinfo -F depends firewire_core
depends:    crc-itu-t
~]# modinfo -F depends crc-itu-t
depends:
```

You have figured out the dependency tree (which does not branch in this example) for the loaded Firewire modules: **firewire_ohci** depends on **firewire_core**, which itself depends on **crc-itu-t**.

You can unload **firewire_ohci** using the **modprobe -v -r <module_name>** command, where **-r** is short for **--remove** and **-v** for **--verbose**:

```
~]# modprobe -r -v firewire_ohci
rmmod /lib/modules/2.6.32-71.el6.x86_64/kernel/drivers/firewire/firewire-ohci.ko
rmmod /lib/modules/2.6.32-71.el6.x86_64/kernel/drivers/firewire/firewire-core.ko
rmmod /lib/modules/2.6.32-71.el6.x86_64/kernel/lib/crc-itu-t.ko
```

The output shows that modules are unloaded in the reverse order that they are loaded, given that no processes depend on any of the modules being unloaded.



IMPORTANT

Although the **rmmod** command can be used to unload kernel modules, it is recommended to use **modprobe -r** instead.

31.5. BLACKLISTING A MODULE

Sometimes, for various performance or security reasons, it is necessary to prevent the system from using a certain kernel module. This can be achieved by *module blacklisting*, which is a mechanism used by the **modprobe** utility to ensure that the kernel cannot automatically load certain modules, or that the modules cannot be loaded at all. This is useful in certain situations, such as when using a certain module poses a security risk to your system, or when the module controls the same hardware or service as another module, and loading both modules would cause the system, or its component, to become unstable or non-operational.

To blacklist a module, you have to add the following line to the specified configuration file in the **/etc/modprobe.d/** directory as root:

```
blacklist <module_name>
```

where **<module_name>** is the name of the module being blacklisted.

You can modify the **/etc/modprobe.d/blacklist.conf** file that already exists on the system by default. However, the preferred method is to create a separate configuration file, **/etc/modprobe.d/<module_name>.conf**, that will contain settings specific only to the given kernel module.

Example 31.4. An example of /etc/modprobe.d/blacklist.conf

```
# 
# Listing a module here prevents the hotplug scripts from loading it.
# Usually that'd be so that some other driver will bind it instead,
# no matter which driver happens to get probed first. Sometimes user
# mode tools can also control driver binding.
#
# Syntax: see modprobe.conf(5).
#

# watchdog drivers
blacklist i8xx_tco

# framebuffer drivers
blacklist aty128fb
```

```

blacklist atyfb
blacklist radeonfb
blacklist i810fb
blacklist cirrusfb
blacklist intelfb
blacklist kyrofb
blacklist i2c-matroxfb
blacklist hgafb
blacklist nvidiafb
blacklist rivafb
blacklist savagefb
blacklist sstfb
blacklist neofb
blacklist tridentfb
blacklist tdfxfb
blacklist virgefb
blacklist vga16fb
blacklist viafb

# ISDN - see bugs 154799, 159068
blacklist hisax
blacklist hisax_fcpcipnp

# sound drivers
blacklist snd-pcsp

# I/O dynamic configuration support for s390x (bz #563228)
blacklist chsc_sch

```

The **blacklist <module_name>** command, however, does not prevent the module from being loaded manually, or from being loaded as a dependency for another kernel module that is not blacklisted. To ensure that a module cannot be loaded on the system at all, modify the specified configuration file in the **/etc/modprobe.d** directory as root with the following line:

install <module_name> /bin/true

where **<module_name>** is the name of the blacklisted module.

Example 31.5. Using module blacklisting as a temporary problem solution

Let's say that a flaw in the Linux kernel's PPP over L2TP module (**pppol2pt**) has been found, and this flaw could be misused to compromise your system. If your system does not require the **pppol2pt** module to function, you can follow this procedure to blacklist **pppol2pt** completely until this problem is fixed:

1. Verify whether **pppol2pt** is currently loaded in the kernel by running the following command:

```

~]# lsmod | grep ^pppol2tp && echo "The module is loaded" || echo "The module is not
loaded"

```

2. If the module is loaded, you need to unload it and all its dependencies to prevent its possible misuse. See [Section 31.4, “Unloading a Module”](#) for instructions on how to safely unload it.
3. Run the following command to ensure that **pppol2pt** cannot be loaded to the kernel:

```
~]# echo "install pppol2tp /bin/true" > /etc/modprobe.d/pppol2tp.conf
```

Note that this command overwrites the content of the **/etc/modprobe.d/pppol2tp.conf** file if it already exists on your system. Check and back up your existing **pppol2tp.conf** before running this command. Also, if you were unable to unload the module, you have to reboot the system for this command to take effect.

After the problem with the **pppol2pt** module has been properly fixed, you can delete the **/etc/modprobe.d/pppol2tp.conf** file or restore its previous content, which will allow your system to load the **pppol2pt** module with its original configuration.



IMPORTANT

Before blacklisting a kernel module, always ensure that the module is not vital for your current system configuration to function properly. Improper blacklisting of a key kernel module can result in an unstable or non-operational system.

31.6. SETTING MODULE PARAMETERS

Like the kernel itself, modules can also take parameters that change their behavior. Most of the time, the default ones work well, but occasionally it is necessary or desirable to set custom parameters for a module. Because parameters cannot be dynamically set for a module that is already loaded into a running kernel, there are two different methods for setting them.

1. Load a kernel module by running the **modprobe** command along with a list of customized parameters on the command line. If the module is already loaded, you need to first unload all its dependencies and the module itself using the **modprobe -r** command. This method allows you to run a kernel module with specific settings without making the changes persistent. See [Section 31.6.1, “Loading a Customized Module – Temporary Changes”](#) for more information.
2. Alternatively, specify a list of the customized parameters in an existing or newly-created file in the **/etc/modprobe.d/** directory. This method ensures that the module customization is persistent by setting the specified parameters accordingly each time the module is loaded, such as after every reboot or **modprobe** command. See [Section 31.6.2, “Loading a Customized Module – Persistent Changes”](#) for more information.

31.6.1. Loading a Customized Module – Temporary Changes

Sometimes it is useful or necessary to run a kernel module temporarily with specific settings. To load a kernel module with customized parameters for the current system session, or until the module is reloaded with different parameters, run **modprobe** in the following format as root:

```
~]# modprobe <module_name> [parameter=value]
```

where **[parameter=value]** represents a list of customized parameters available to that module. When loading a module with custom parameters on the command line, be aware of the following:

- You can enter multiple parameters and values by separating them with spaces.
- Some module parameters expect a list of comma-separated values as their argument. When entering the list of values, do not insert a space after each comma, or **modprobe** will incorrectly interpret the values following spaces as additional parameters.

- The **modprobe** command silently succeeds with an exit status of **0** if it successfully loads the module, or the module is *already* loaded into the kernel. Thus, you must ensure that the module is not already loaded before attempting to load it with custom parameters. The **modprobe** command does not automatically reload the module, or alert you that it is already loaded.

The following procedure illustrates the recommended steps to load a kernel module with custom parameters on the **e1000e** module, which is the network driver for Intel PRO/1000 network adapters, as an example:

Procedure 31.1. Loading a Kernel Module with Custom Parameters

- Verify whether the module is not already loaded into the kernel by running the following command:

```
~]# lsmod|grep e1000e
e1000e      236338  0
ptp          9614  1 e1000e
```

Note that the output of the command in this example indicates that the **e1000e** module is already loaded into the kernel. It also shows that this module has one dependency, the **ptp** module.

- If the module is already loaded into the kernel, you must unload the module and all its dependencies before proceeding with the next step. See [Section 31.4, “Unloading a Module”](#) for instructions on how to safely unload it.
- Load the module and list all custom parameters after the module name. For example, if you wanted to load the Intel PRO/1000 network driver with the interrupt throttle rate set to 3000 interrupts per second for the first, second and third instances of the driver, and Energy Efficient Ethernet (EEE) turned on [5], you would run, as root:

```
~]# modprobe e1000e InterruptThrottleRate=3000,3000,3000 EEE=1
```

This example illustrates passing multiple values to a single parameter by separating them with commas and omitting any spaces between them.

31.6.2. Loading a Customized Module - Persistent Changes

If you want to ensure that a kernel module is always loaded with specific settings, modify an existing or newly-created file in the **/etc/modprobe.d/** directory with a line in the following format.

```
~]# options <module_name> [parameter=value]
```

where **[parameter=value]** represents a list of customized parameters available to that module.

The following procedure illustrates the recommended steps for loading a kernel module with custom parameters on the **b43** module for Open Firmware for wireless networks, ensuring that changes persist between module reloads.

Procedure 31.2. Loading a Kernel Module with Custom Parameters - Persistent Changes

- Add the following line to the **/etc/modprobe.d/openfwf.conf** file, which ensures that the **b43** module is always loaded with QoS and hardware-accelerated cryptography disabled:

```
options b43 nohwcrypt=1 qos=0
```

- Verify whether the module is not already loaded into the kernel by running the following command:

```
~]# lsmod|grep ^b43  
~]#
```

Note that the output of the command in this example indicates that the module is currently not loaded into the kernel.

- If the module is already loaded into the kernel, you must unload the module and all its dependencies before proceeding with the next step. See [Section 31.4, “Unloading a Module”](#) for instructions on how to safely unload it.
- Load the **b43** module by running the following command:

```
~]# modprobe b43
```

31.7. PERSISTENT MODULE LOADING

As shown in [Example 31.1, “Listing information about a kernel module with lsmod”](#), many kernel modules are loaded automatically at boot time. You can specify additional modules to be loaded by creating a new **<file_name>.modules** file in the **/etc/sysconfig/modules/** directory, where **<file_name>** is any descriptive name of your choice. Your **<file_name>.modules** files are treated by the system startup scripts as shell scripts, and as such should begin with an *interpreter directive* (also called a “bang line”) as their first line:

Example 31.6. First line of a **<file_name>.modules** file

```
#!/bin/sh
```

Additionally, the **<file_name>.modules** file should be executable. You can make it executable by running:

```
modules]# chmod +x <file_name>.modules
```

For example, the following **bluez-uinput.modules** script loads the **uinput** module:

Example 31.7. **/etc/sysconfig/modules/bluez-uinput.modules**

```
#!/bin/sh  
if [ ! -c /dev/input/uinput ] ; then  
    exec /sbin/modprobe uinput >/dev/null 2>&1  
fi
```

The **if**-conditional statement on the third line ensures that the **/dev/input/uinput** file does *not* already exist (the **!** symbol negates the condition), and, if that is the case, loads the **uinput** module by calling **exec /sbin/modprobe uinput**. Note that the **uinput** module creates the **/dev/input/uinput**

file, so testing to see if that file exists serves as verification of whether the **uinput** module is loaded into the kernel.

The following `>/dev/null 2>&1` clause at the end of that line redirects any output to `/dev/null` so that the **modprobe** command remains quiet.

31.8. SPECIFIC KERNEL MODULE CAPABILITIES

This section explains how to enable specific kernel capabilities using various kernel modules.

31.8.1. Using Channel Bonding

Red Hat Enterprise Linux allows administrators to bind NICs together into a single channel using the **bonding** kernel module and a special network interface, called a *channel bonding interface*. Channel bonding enables two or more network interfaces to act as one, simultaneously increasing the bandwidth and providing redundancy.

To channel bond multiple network interfaces, the administrator must perform the following steps:

1. Configure a channel bonding interface as outlined in [Section 11.2.4, “Channel Bonding Interfaces”](#).
2. To enhance performance, adjust available module options to ascertain what combination works best. Pay particular attention to the **miimon** or **arp_interval** and the **arp_ip_target** parameters. See [Section 31.8.1.1, “Bonding Module Directives”](#) for a list of available options and how to quickly determine the best ones for your bonded interface.

31.8.1.1. Bonding Module Directives

It is a good idea to test which channel bonding module parameters work best for your bonded interfaces before adding them to the **BONDING_OPTS="<bonding parameters>"** directive in your bonding interface configuration file (**ifcfg-bond0** for example). Parameters to bonded interfaces can be configured without unloading (and reloading) the bonding module by manipulating files in the **sysfs** file system.

sysfs is a virtual file system that represents kernel objects as directories, files and symbolic links. **sysfs** can be used to query for information about kernel objects, and can also manipulate those objects through the use of normal file system commands. The **sysfs** virtual file system has a line in **/etc/fstab**, and is mounted under the **/sys** directory. All bonding interfaces can be configured dynamically by interacting with and manipulating files under the **/sys/class/net** directory.

In order to determine the best parameters for your bonding interface, create a channel bonding interface file such as **ifcfg-bond0** by following the instructions in [Section 11.2.4, “Channel Bonding Interfaces”](#). Insert the **SLAVE=yes** and **MASTER=bond0** directives in the configuration files for each interface bonded to bond0. Once this is completed, you can proceed to testing the parameters.

First, bring up the bond you created by running **ifconfig bond<N> up** as root:

```
~]# ifconfig bond0 up
```

If you have correctly created the **ifcfg-bond0** bonding interface file, you will be able to see **bond0** listed in the output of running **ifconfig** (without any options):

```
~]# ifconfig
bond0  Link encap:Ethernet HWaddr 00:00:00:00:00:00
        UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500 Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)
eth0    Link encap:Ethernet HWaddr 52:54:00:26:9E:F1
        inet addr:192.168.122.251 Bcast:192.168.122.255 Mask:255.255.255.0
        inet6 addr: fe80::5054:ff:fe26:9ef1/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:207 errors:0 dropped:0 overruns:0 frame:0
        TX packets:205 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:70374 (68.7 KiB) TX bytes:25298 (24.7 KiB)
[output truncated]
```

To view all existing bonds, even if they are not up, run:

```
~]# cat /sys/class/net/bonding_masters
bond0
```

You can configure each bond individually by manipulating the files located in the **/sys/class/net/bond<N>/bonding** directory. First, the bond you are configuring must be taken down:

```
~]# ifconfig bond0 down
```

As an example, to enable MII monitoring on bond0 with a 1 second interval, you could run (as root):

```
~]# echo 1000 > /sys/class/net/bond0/bonding/miimon
```

To configure bond0 for ***balance-alb*** mode, you could run either:

```
~]# echo 6 > /sys/class/net/bond0/bonding	mode
```

...or, using the name of the mode:

```
~]# echo balance-alb > /sys/class/net/bond0/bonding/mode
```

After configuring options for the bond in question, you can bring it up and test it by running **ifconfig bond<N> up**. If you decide to change the options, take the interface down, modify its parameters using **sysfs**, bring it back up, and re-test.

Once you have determined the best set of parameters for your bond, add those parameters as a space-separated list to the **BONDING_OPTS=** directive of the **/etc/sysconfig/network-scripts/ifcfg-bond<N>** file for the bonding interface you are configuring. Whenever that bond is brought up (for example, by the system during the boot sequence if the **ONBOOT=yes** directive is set), the bonding options specified in the **BONDING_OPTS** will take effect for that bond. For more information on configuring bonding interfaces (and **BONDING_OPTS**), see [Section 11.2.4, “Channel Bonding Interfaces”](#).

The following list provides the names of many of the more common channel bonding parameters, along with a descriptions of what they do. For more information, see the brief descriptions for each **parm** in **modinfo bonding** output, or the exhaustive descriptions in the **bonding.txt** file in the kernel-doc

package (see [Section 31.9, “Additional Resources”](#)).

Bonding Interface Parameters

arp_interval=<time_in_milliseconds>

Specifies (in milliseconds) how often ARP monitoring occurs. When configuring this setting, a good starting point for this parameter is **1000**.



IMPORTANT

It is essential that both **arp_interval** and **arp_ip_target** parameters are specified, or, alternatively, the **miimon** parameter is specified. Failure to do so can cause degradation of network performance in the event that a link fails.

If using this setting while in **mode=0** or **mode=2** (the two load-balancing modes), the network switch must be configured to distribute packets evenly across the NICs. For more information on how to accomplish this, see the **bonding.txt** file in the **kernel-doc** package (see [Section 31.9, “Additional Resources”](#)).

The value is set to **0** by default, which disables it.

arp_ip_target=<ip_address>[,<ip_address_2>,...<ip_address_16>]

Specifies the target IP address of ARP requests when the **arp_interval** parameter is enabled. Up to 16 IP addresses can be specified in a comma separated list.

arp_validate=<value>

Validate source/distribution of ARP probes; default is **none**. Other valid values are **active**, **backup**, and **all**.

downdelay=<time_in_milliseconds>

Specifies (in milliseconds) how long to wait after link failure before disabling the link. The value must be a multiple of the value specified in the **miimon** parameter. The value is set to **0** by default, which disables it.

lacp_rate=<value>

Specifies the rate at which link partners should transmit LACPDU packets in 802.3ad mode. Possible values are:

- **slow** or **0** – Default setting. This specifies that partners should transmit LACPDUs every 30 seconds.
- **fast** or **1** – Specifies that partners should transmit LACPDUs every 1 second.

miimon=<time_in_milliseconds>

Specifies (in milliseconds) how often MII link monitoring occurs. This is useful if high availability is required because MII is used to verify that the NIC is active. To verify that the driver for a particular NIC supports the MII tool, type the following command as root:

```
~]# ethtool <interface_name> | grep "Link detected:"
```

In this command, replace <interface_name> with the name of the device interface, such as **eth0**, not the bond interface. If MII is supported, the command returns:

Link detected: yes

If using a bonded interface for high availability, the module for each NIC must support MII. Setting the value to **0** (the default), turns this feature off. When configuring this setting, a good starting point for this parameter is **100**.



IMPORTANT

It is essential that both **arp_interval** and **arp_ip_target** parameters are specified, or, alternatively, the **miimon** parameter is specified. Failure to do so can cause degradation of network performance in the event that a link fails.

mode=<value>

Allows you to specify the bonding policy. The <value> can be one of:

- **balance-rr** or **0** – Sets a round-robin policy for fault tolerance and load balancing. Transmissions are received and sent out sequentially on each bonded slave interface beginning with the first one available.
- **active-backup** or **1** – Sets an active-backup policy for fault tolerance. Transmissions are received and sent out via the first available bonded slave interface. Another bonded slave interface is only used if the active bonded slave interface fails.
- **balance-xor** or **2** – Sets an XOR (exclusive-or) policy for fault tolerance and load balancing. Using this method, the interface matches up the incoming request's MAC address with the MAC address for one of the slave NICs. Once this link is established, transmissions are sent out sequentially beginning with the first available interface.
- **broadcast** or **3** – Sets a broadcast policy for fault tolerance. All transmissions are sent on all slave interfaces.
- **802.3ad** or **4** – Sets an IEEE 802.3ad dynamic link aggregation policy. Creates aggregation groups that share the same speed and duplex settings. Transmits and receives on all slaves in the active aggregator. Requires a switch that is 802.3ad compliant.
- **balance-tlb** or **5** – Sets a Transmit Load Balancing (TLB) policy for fault tolerance and load balancing. The outgoing traffic is distributed according to the current load on each slave interface. Incoming traffic is received by the current slave. If the receiving slave fails, another slave takes over the MAC address of the failed slave. This mode is only suitable for local addresses known to the kernel bonding module and therefore cannot be used behind a bridge with virtual machines.
- **balance-alb** or **6** – Sets an Adaptive Load Balancing (ALB) policy for fault tolerance and load balancing. Includes transmit and receive load balancing for **IPv4** traffic. Receive load balancing is achieved through **ARP** negotiation. This mode is only suitable for local addresses known to the kernel bonding module and therefore cannot be used behind a bridge with virtual machines.

num_unsol_na=<number>

Specifies the number of unsolicited IPv6 Neighbor Advertisements to be issued after a failover event. One unsolicited NA is issued immediately after the failover.

The valid range is **0 - 255**; the default value is **1**. This parameter affects only the active-backup mode.

primary=<interface_name>

Specifies the interface name, such as **eth0**, of the primary device. The **primary** device is the first of the bonding interfaces to be used and is not abandoned unless it fails. This setting is particularly useful when one NIC in the bonding interface is faster and, therefore, able to handle a bigger load.

This setting is only valid when the bonding interface is in **active-backup** mode. See the **bonding.txt** file in the kernel-doc package (see [Section 31.9, “Additional Resources”](#)).

primary_reselect=<value>

Specifies the reselection policy for the primary slave. This affects how the primary slave is chosen to become the active slave when failure of the active slave or recovery of the primary slave occurs. This parameter is designed to prevent flip-flopping between the primary slave and other slaves. Possible values are:

- **always** or **0** (default) – The primary slave becomes the active slave whenever it comes back up.
- **better** or **1** – The primary slave becomes the active slave when it comes back up, if the speed and duplex of the primary slave is better than the speed and duplex of the current active slave.
- **failure** or **2** – The primary slave becomes the active slave only if the current active slave fails and the primary slave is up.

The **primary_reselect** setting is ignored in two cases:

- If no slaves are active, the first slave to recover is made the active slave.
- When initially enslaved, the primary slave is always made the active slave.

Changing the **primary_reselect** policy via **sysfs** will cause an immediate selection of the best active slave according to the new policy. This may or may not result in a change of the active slave, depending upon the circumstances

updelay=<time_in_milliseconds>

Specifies (in milliseconds) how long to wait before enabling a link. The value must be a multiple of the value specified in the **miimon** parameter. The value is set to **0** by default, which disables it.

use_carrier=<number>

Specifies whether or not **miimon** should use MII/ETHTOOL ioctls or **netif_carrier_ok()** to determine the link state. The **netif_carrier_ok()** function relies on the device driver to maintain its state with **netif_carrier_on/off**; most device drivers support this function.

The MII/ETHROOL ioctls tools utilize a deprecated calling sequence within the kernel. However, this is still configurable in case your device driver does not support **netif_carrier_on/off**.

Valid values are:

- **1** – Default setting. Enables the use of **netif_carrier_ok()**.
- **0** – Enables the use of MII/ETHTOOL ioctls.



NOTE

If the bonding interface insists that the link is up when it should not be, it is possible that your network device driver does not support **netif_carrier_on/off**.

xmit_hash_policy=<value>

Selects the transmit hash policy used for slave selection in **balance-xor** and **802.3ad** modes. Possible values are:

- **0 or layer2** – Default setting. This parameter uses the XOR of hardware MAC addresses to generate the hash. The formula used is:

```
(<source_MAC_address> XOR <destination_MAC>) MODULO <slave_count>
```

This algorithm will place all traffic to a particular network peer on the same slave, and is 802.3ad compliant.

- **1 or layer3+4** – Uses upper layer protocol information (when available) to generate the hash. This allows for traffic to a particular network peer to span multiple slaves, although a single connection will not span multiple slaves.

The formula for unfragmented TCP and UDP packets used is:

```
((<source_port> XOR <dest_port>) XOR  
(<source_IP> XOR <dest_IP>) AND 0xffff)  
MODULO <slave_count>
```

For fragmented TCP or UDP packets and all other IP protocol traffic, the source and destination port information is omitted. For non-IP traffic, the formula is the same as the **layer2** transmit hash policy.

This policy intends to mimic the behavior of certain switches; particularly, Cisco switches with PFC2 as well as some Foundry and IBM products.

The algorithm used by this policy is not 802.3ad compliant.

- **2 or layer2+3** – Uses a combination of layer2 and layer3 protocol information to generate the hash.

Uses XOR of hardware MAC addresses and IP addresses to generate the hash. The formula is:

```
(((<source_IP> XOR <dest_IP>) AND 0xffff) XOR  
(<source_MAC> XOR <destination_MAC>))  
MODULO <slave_count>
```

This algorithm will place all traffic to a particular network peer on the same slave. For non-IP traffic, the formula is the same as for the layer2 transmit hash policy.

This policy is intended to provide a more balanced distribution of traffic than layer2 alone, especially in environments where a layer3 gateway device is required to reach most destinations.

This algorithm is 802.3ad compliant.

31.9. ADDITIONAL RESOURCES

For more information on kernel modules and their utilities, see the following resources.

Installed Documentation

- **lsmod(8)** – The manual page for the **lsmod** command.
- **modinfo(8)** – The manual page for the **modinfo** command.
- **modprobe(8)**⁵ – The manual page for the **modprobe** command.
- **rmmod(8)** – The manual page for the **rmmod** command.
- **ethtool(8)** – The manual page for the **ethtool** command.
- **mii-tool(8)** – The manual page for the **mii-tool** command.

Installable Documentation

- **/usr/share/doc/kernel-doc-<kernel_version>/Documentation** – This directory, which is provided by the kernel-doc package, contains information on the kernel, kernel modules, and their respective parameters. Before accessing the kernel documentation, you must run the following command as root:

```
~]# yum install kernel-doc
```

Online Documentation

- – The Red Hat Knowledgebase article [Which bonding modes work when used with a bridge that virtual machine guests connect to?](#)

[5] Despite what the example might imply, Energy Efficient Ethernet is turned on by default in the **e1000e** driver.

CHAPTER 32. THE KDUMP CRASH RECOVERY SERVICE

When the **kdump** crash dumping mechanism is enabled, the system is booted from the context of another kernel. This second kernel reserves a small amount of memory and its only purpose is to capture the core dump image in case the system crashes.

Being able to analyze the core dump significantly helps to determine the exact cause of the system failure, and it is therefore strongly recommended to have this feature enabled. This chapter explains how to configure, test, and use the **kdump** service in Red Hat Enterprise Linux, and provides a brief overview of how to analyze the resulting core dump using the **crash** debugging utility.

32.1. INSTALLING THE KDUMP SERVICE

In order to use the **kdump** service on your system, make sure you have the `kexec-tools` package installed. To do so, type the following at a shell prompt as **root**:

```
~]# yum install kexec-tools
```

For more information on how to install new packages in Red Hat Enterprise Linux, see [Section 8.2.4, “Installing Packages”](#).

32.2. CONFIGURING THE KDUMP SERVICE

There are three common means of configuring the **kdump** service: at the first boot, using the **Kernel Dump Configuration** graphical utility, and doing so manually on the command line.



IMPORTANT

A limitation in the current implementation of the **Intel IOMMU** driver can occasionally prevent the **kdump** service from capturing the core dump image. To use **kdump** on Intel architectures reliably, it is advised that the IOMMU support is disabled.



WARNING

It is known that the **kdump** service does not work reliably on certain combinations of HP Smart Array devices and system boards from the same vendor. Consequent to this, users are strongly advised to test the configuration before using it in production environment, and if necessary, configure **kdump** to store the kernel crash dump to a remote machine over a network. For more information on how to test the **kdump** configuration, see [Section 32.2.4, “Testing the Configuration”](#).

32.2.1. Configuring kdump at First Boot

When the system boots for the first time, the **firstboot** application is launched to guide the user through the initial configuration of the freshly installed system. To configure **kdump**, navigate to the **Kdump** section and follow the instructions below.

1. Select the **Enable kdump?** check box to allow the **kdump** daemon to start at boot time. This will enable the service for runlevels **2, 3, 4**, and **5**, and start it for the current session. Similarly, unselecting the check box will disable it for all runlevels and stop the service immediately.
2. Click the up and down arrow buttons next to the **Kdump Memory** field to increase or decrease the value to configure the amount of memory that is reserved for the **kdump** kernel. Notice that the **Usable System Memory** field changes accordingly showing you the remaining memory that will be available to the system.



IMPORTANT

This section is available only if the system has enough memory. To learn about minimum memory requirements of the Red Hat Enterprise Linux 6 system, read the *Required minimums* section of the [Red Hat Enterprise Linux Technology Capabilities and Limits](#) comparison chart. When the **kdump** crash recovery is enabled, the minimum memory requirements increase by the amount of memory reserved for it. This value is determined by the user, and defaults to 128 MB plus 64 MB for each TB of physical memory (that is, a total of 192 MB for a system with 1 TB of physical memory). The memory can be attempted up to the maximum of 896 MB if required. This is recommended especially in large environments, for example in systems with a large number of Logical Unit Numbers (LUNs).

32.2.2. Using the Kernel Dump Configuration Utility

To start the **Kernel Dump Configuration** utility, select **System** → **Administration** → **Kernel crash dumps** from the panel, or type **system-config-kdump** at a shell prompt. You will be presented with a window as shown in [Figure 32.1, “Basic Settings”](#).

The utility allows you to configure **kdump** as well as to enable or disable starting the service at boot time. When you are done, click **Apply** to save the changes. The system reboot will be requested, and unless you are already authenticated, you will be prompted to enter the superuser password.



IMPORTANT

On IBM System z or PowerPC systems with SELinux running in Enforcing mode, the **kdumpgui_run_bootloader** Boolean must be enabled before launching the **Kernel Dump Configuration** utility. This Boolean allows **system-config-kdump** to run the boot loader in the **bootloader_t** SELinux domain. To permanently enable the Boolean, run the following command as **root**:

```
~]# setsebool -P kdumpgui_run_bootloader 1
```

Enabling the Service

To start the **kdump** daemon at boot time, click the **Enable** button on the toolbar. This will enable the service for runlevels **2, 3, 4**, and **5**, and start it for the current session. Similarly, clicking the **Disable** button will disable it for all runlevels and stop the service immediately.

For more information on runlevels and configuring services in general, see [Chapter 12, Services and Daemons](#).

The Basic Settings Tab

The **Basic Settings** tab enables you to configure the amount of memory that is reserved for the **kdump** kernel. To do so, select the **Manual kdump memory settings** radio button, and click the up and down arrow buttons next to the **New kdump Memory** field to increase or decrease the value. Notice that the

Usable Memory field changes accordingly showing you the remaining memory that will be available to the system.

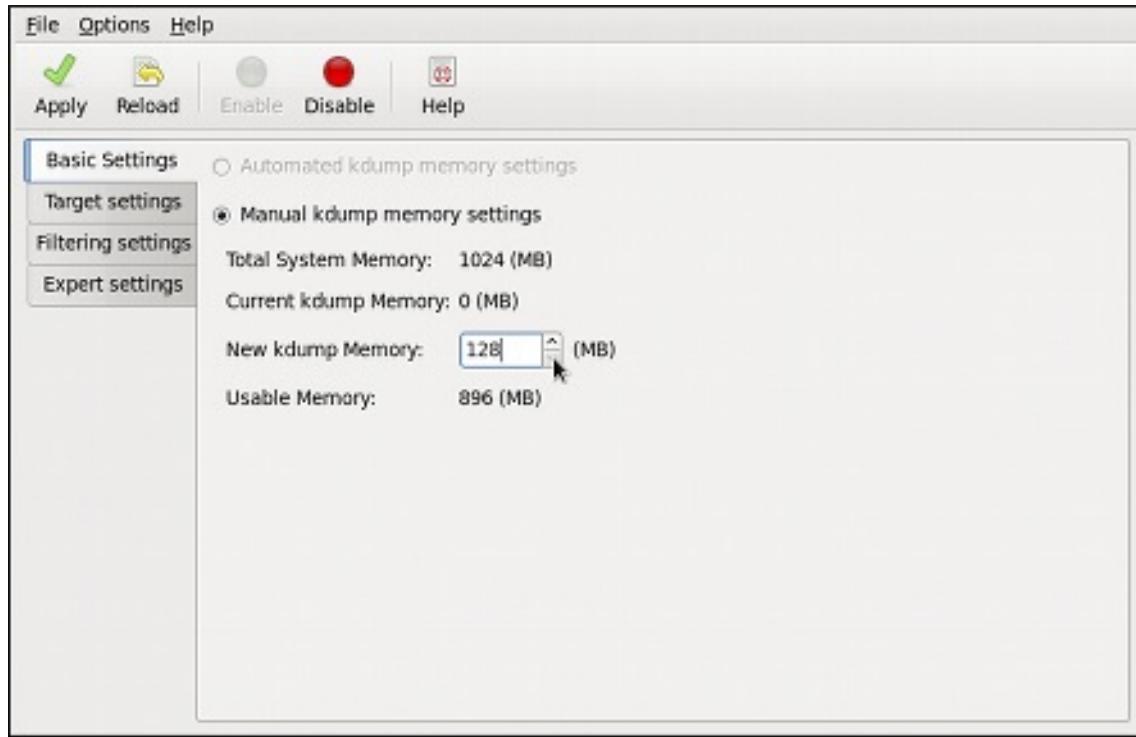


Figure 32.1. Basic Settings



IMPORTANT

This section is available only if the system has enough memory. To learn about minimum memory requirements of the Red Hat Enterprise Linux 6 system, read the *Required minimums* section of the [Red Hat Enterprise Linux Technology Capabilities and Limits](#) comparison chart. When the **kdump** crash recovery is enabled, the minimum memory requirements increase by the amount of memory reserved for it. This value is determined by the user, and defaults to 128 MB plus 64 MB for each TB of physical memory (that is, a total of 192 MB for a system with 1 TB of physical memory). The memory can be attempted up to the maximum of 896 MB if required. This is recommended especially in large environments, for example in systems with a large number of Logical Unit Numbers (LUNs).

The Target Settings Tab

The **Target Settings** tab enables you to specify the target location for the **vmcore** dump. It can be either stored as a file in a local file system, written directly to a device, or sent over a network using the NFS (Network File System) or SSH (Secure Shell) protocol.

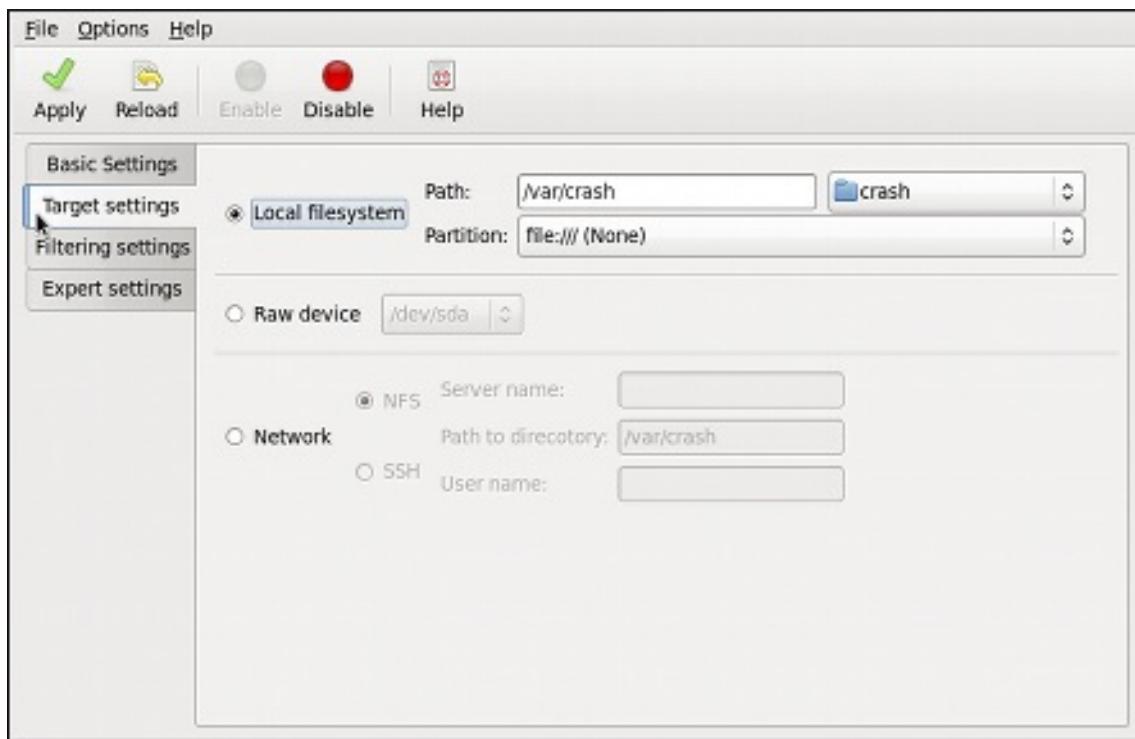


Figure 32.2. Target Settings

To save the dump to the local file system, select the **Local filesystem** radio button. Optionally, you can customize the settings by choosing a different partition from the **Partition**, and a target directory from the **Path** pulldown lists.

To write the dump directly to a device, select the **Raw device** radio button, and choose the desired target device from the pulldown list next to it.

To store the dump to a remote machine, select the **Network** radio button. To use the NFS protocol, select the **NFS** radio button, and fill the **Server name** and **Path to directory** fields. To use the SSH protocol, select the **SSH** radio button, and fill the **Server name**, **Path to directory**, and **User name** fields with the remote server address, target directory, and a valid remote user name respectively. See [Chapter 14, OpenSSH](#) for information on how to configure an SSH server, and how to set up a key-based authentication.



NOTE

When using Direct-Access Storage Devices (DASDs) as the kdump target, the devices must be specified in the **/etc/dasd.conf** file with other DASDs, for example:

```
0.0.2098
0.0.2198
0.0.2298
0.0.2398
```

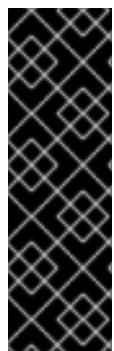
Where **0.0.2298** and **0.0.2398** are the DASDs used as the kdump target.

Similarly, when using FCP-attached Small Computer System Interface (SCSI) disks as the kdump target, the disks must be specified in the **/etc/zfcp.conf** file with other FCP-Attached SCSI disks, for example:

```
0.0.3d0c 0x500507630508c1ae 0x402424aa00000000
0.0.3d0c 0x500507630508c1ae 0x402424ab00000000
0.0.3d0c 0x500507630508c1ae 0x402424ac00000000
```

Where **0.0.3d0c 0x500507630508c1ae 0x402424ab00000000** and **0.0.3d0c 0x500507630508c1ae 0x402424ac00000000** are the FCP-attached SCSI disks used as the kdump target.

See the [Adding DASDs](#) and [Adding FCP-Attached Logical Units \(LUNs\)](#) chapters in the *Installation Guide* for Red Hat Enterprise Linux 6 for detailed information about configuring DASDs and FCP-attached SCSI disks.



IMPORTANT

When transferring a core file to a remote target over SSH, the core file needs to be serialized for the transfer. This creates a **vmcore.flat** file in the **/var/crash/** directory on the target system, which is unreadable by the **crash** utility. To convert **vmcore.flat** to a dump file that is readable by **crash**, run the following command as root on the target system:

```
~]# /usr/sbin/makedumpfile -R */tmp/vmcore-rearranged* < *vmcore.flat*
```

For a complete list of currently supported targets, see [Table 32.1, “Supported kdump targets”](#).

Table 32.1. Supported kdump targets

Type	Supported Targets	Unsupported Targets
Raw device	All locally attached raw disks and partitions.	—
Local file system	ext2, ext3, ext4, minix, btrfs and xfs file systems on directly attached disk drives, hardware RAID logical drives, LVM devices, and mdraid arrays.	Any local file system not explicitly listed as supported in this table, including the auto type (automatic file system detection).

Type	Supported Targets	Unsupported Targets
Remote directory	Remote directories accessed using the NFS or SSH protocol over IPv4 .	Remote directories on the rootfs file system accessed using the NFS protocol.
iSCSI	Remote directories accessed using the iSCSI protocol over software initiators, unless iBFT (<i>iSCSI Boot Firmware Table</i>) is utilized.	Remote directories accessed using the iSCSI protocol using iBFT .
Multipath	Multipath-based storage. ^[a]	Remote directories accessed using the iSCSI protocol over hardware initiators.
—	—	Remote directories accessed over IPv6 .
.	.	Remote directories accessed using the SMB/CIFS protocol.
.	.	Remote directories accessed using the FCoE (<i>Fibre Channel over Ethernet</i>) protocol.
.	.	Remote directories accessed using wireless network interfaces.

[a] Supported in Red Hat Enterprise Linux 6 from kexec-tools-2.0.0-245.el6 onwards.

The Filtering Settings Tab

The **Filtering Settings** tab enables you to select the filtering level for the **vmcore** dump.

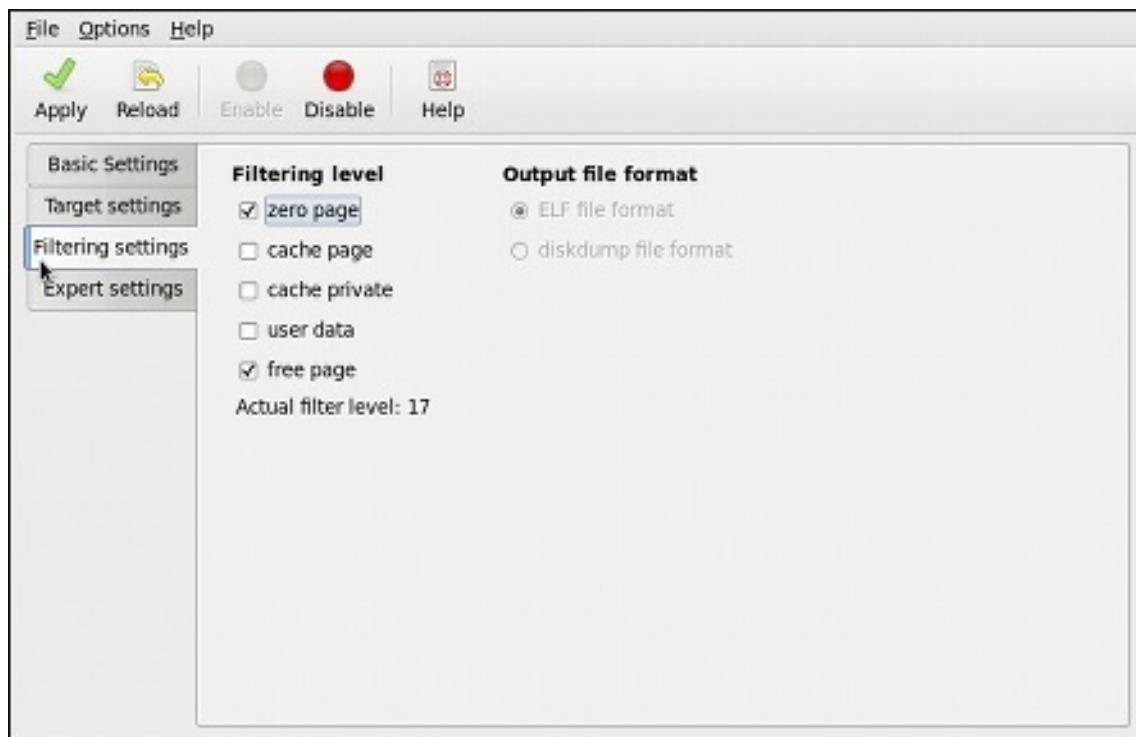


Figure 32.3. Filtering Settings

To exclude the **zero page**, **cache page**, **cache private**, **user data**, or **free page** from the dump, select the check box next to the appropriate label.

The Expert Settings Tab

The **Expert Settings** tab enables you to choose which kernel and initial RAM disk to use, as well as to customize the options that are passed to the kernel and the core collector program.

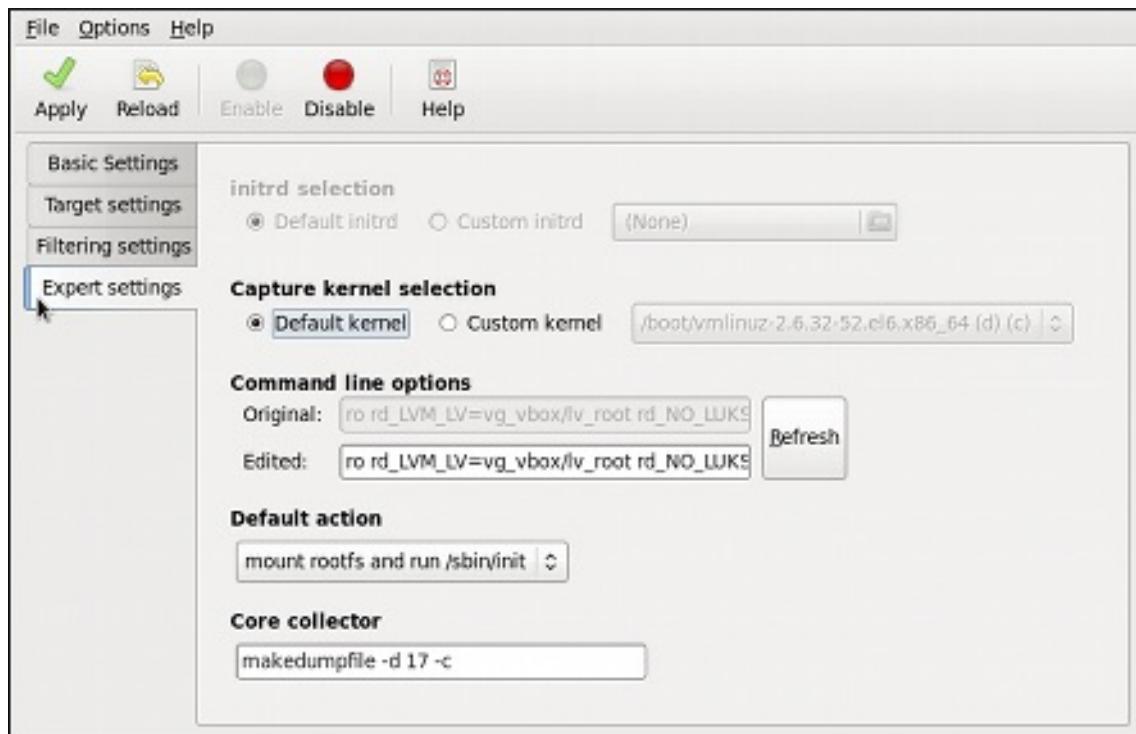


Figure 32.4. Expert Settings

To use a different initial RAM disk, select the **Custom initrd** radio button, and choose the desired RAM disk from the pulldown list next to it.

To capture a different kernel, select the **Custom kernel** radio button, and choose the desired kernel image from the pulldown list on the right.

To adjust the list of options that are passed to the kernel at boot time, edit the content of the **Edited** text field. Note that you can always revert your changes by clicking the **Refresh** button.

To choose what action to perform when **kdump** fails to create a core dump, select an appropriate option from the **Default action** pulldown list. Available options are **mount rootfs and run /sbin/init** (the default action), **reboot** (to reboot the system), **shell** (to present a user with an interactive shell prompt), **halt** (to halt the system), and **poweroff** (to power the system off).

To customize the options that are passed to the **makedumpfile** core collector, edit the **Core collector** text field; see [the section called “Configuring the Core Collector”](#) for more information.

32.2.3. Configuring kdump on the Command Line

Configuring the Memory Usage

Memory reserved for the kdump kernel is always reserved during system boot, which means that the amount of memory is specified in the system's boot loader configuration. This section will explain how to change the amount of reserved memory on AMD64 and Intel 64 systems and IBM Power Systems servers using the **GRUB** boot loader, and on IBM System z using **zipl**. To configure the amount of memory to be reserved for the **kdump** kernel, edit the **/boot/grub/grub.conf** file and add **crashkernel=<size>M** or **crashkernel=auto** to the list of kernel options as shown in [Example 32.1, “A sample /boot/grub/grub.conf file”](#). Note that the **crashkernel=auto** option only reserves the memory if the physical memory of the system is equal to or greater than:

- **2 GB** on 32-bit and 64-bit **x86** architectures;
- **2 GB** on **PowerPC** if the page size is 4 KB, or **8 GB** otherwise;
- **4 GB** on **IBM S/390**.

Example 32.1. A sample /boot/grub/grub.conf file

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE: You have a /boot partition. This means that
#         all kernel and initrd paths are relative to /boot/, eg.
#         root (hd0,0)
#         kernel /vmlinuz-version ro root=/dev/sda3
#         initrd /initrd
#boot=/dev/sda
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (2.6.32-220.el6.x86_64)
root (hd0,0)
kernel /vmlinuz-2.6.32-220.el6.x86_64 ro root=/dev/sda3 crashkernel=128M
initrd /initramfs-2.6.32-220.el6.x86_64.img
```



IMPORTANT

This section is available only if the system has enough memory. To learn about minimum memory requirements of the Red Hat Enterprise Linux 6 system, read the *Required minimums* section of the [Red Hat Enterprise Linux Technology Capabilities and Limits](#) comparison chart. When the **kdump** crash recovery is enabled, the minimum memory requirements increase by the amount of memory reserved for it. This value is determined by the user, and defaults to 128 MB plus 64 MB for each TB of physical memory (that is, a total of 192 MB for a system with 1 TB of physical memory). The memory can be attempted up to the maximum of 896 MB if required. This is recommended especially in large environments, for example in systems with a large number of Logical Unit Numbers (LUNs).

Configuring the Target Type

When a kernel crash is captured, the core dump can be either stored as a file in a local file system, written directly to a device, or sent over a network using the NFS (Network File System) or SSH (Secure Shell) protocol. Only one of these options can be set at the moment, and the default option is to store the **vmcore** file in the **/var/crash/** directory of the local file system. To change this, as **root**, open the **/etc/kdump.conf** configuration file in a text editor and edit the options as described below.

To change the local directory in which the core dump is to be saved, remove the hash sign ("#") from the beginning of the **#path /var/crash** line, and replace the value with a desired directory path. Optionally, if you want to write the file to a different partition, follow the same procedure with the **#ext4 /dev/sda3** line as well, and change both the file system type and the device (a device name, a file system label, and UUID are all supported) accordingly. For example:

```
ext3 /dev/sda4
      path /usr/local/cores
```

To write the dump directly to a device, remove the hash sign ("#") from the beginning of the **#raw /dev/sda5** line, and replace the value with a desired device name. For example:

```
raw /dev/sdb1
```

To store the dump to a remote machine using the NFS protocol, remove the hash sign ("#") from the beginning of the **#net my.server.com:/export/tmp** line, and replace the value with a valid host name and directory path. For example:

```
net penguin.example.com:/export/cores
```

To store the dump to a remote machine using the SSH protocol, remove the hash sign ("#") from the beginning of the **#net user@my.server.com** line, and replace the value with a valid user name and host name. For example:

```
net john@penguin.example.com
```

See [Chapter 14, OpenSSH](#) for information on how to configure an SSH server, and how to set up a key-based authentication.

For a complete list of currently supported targets, see [Table 32.1, “Supported kdump targets”](#).



NOTE

When using Direct-Access Storage Devices (DASDs) as the kdump target, the devices must be specified in the **/etc/dasd.conf** file with other DASDs, for example:

```
0.0.2098
0.0.2198
0.0.2298
0.0.2398
```

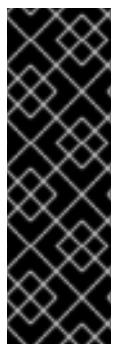
Where **0.0.2298** and **0.0.2398** are the DASDs used as the kdump target.

Similarly, when using FCP-attached Small Computer System Interface (SCSI) disks as the kdump target, the disks must be specified in the **/etc/zfcp.conf** file with other FCP-Attached SCSI disks, for example:

```
0.0.3d0c 0x500507630508c1ae 0x402424aa00000000
0.0.3d0c 0x500507630508c1ae 0x402424ab00000000
0.0.3d0c 0x500507630508c1ae 0x402424ac00000000
```

Where **0.0.3d0c 0x500507630508c1ae 0x402424ab00000000** and **0.0.3d0c 0x500507630508c1ae 0x402424ac00000000** are the FCP-attached SCSI disks used as the kdump target.

See the [Adding DASDs](#) and [Adding FCP-Attached Logical Units \(LUNs\)](#) chapters in the *Installation Guide* for Red Hat Enterprise Linux 6 for detailed information about configuring DASDs and FCP-attached SCSI disks.



IMPORTANT

When transferring a core file to a remote target over SSH, the core file needs to be serialized for the transfer. This creates a **vmcore.flat** file in the **/var/crash/** directory on the target system, which is unreadable by the **crash** utility. To convert **vmcore.flat** to a dump file that is readable by **crash**, run the following command as **root** on the target system:

```
~]# /usr/sbin/makedumpfile -R */tmp/vmcore-rearranged* < *vmcore.flat*
```

Configuring the Core Collector

To reduce the size of the **vmcore** dump file, **kdump** allows you to specify an external application (that is, a core collector) to compress the data, and optionally leave out all irrelevant information. Currently, the only fully supported core collector is **makedumpfile**.

To enable the core collector, as **root**, open the **/etc/kdump.conf** configuration file in a text editor, remove the hash sign ("#") from the beginning of the **#core_collector makedumpfile -c --message-level 1 -d 31** line, and edit the command-line options as described below.

To enable the dump file compression, add the **-c** parameter. For example:

```
core_collector makedumpfile -c
```

To remove certain pages from the dump, add the **-d value** parameter, where *value* is a sum of values of pages you want to omit as described in [Table 32.2, "Supported filtering levels"](#). For example, to remove both zero and free pages, use the following:

```
core_collector makedumpfile -d 17 -c
```

See the manual page for **makedumpfile** for a complete list of available options.

Table 32.2. Supported filtering levels

Option	Description
1	Zero pages
2	Cache pages
4	Cache private
8	User pages
16	Free pages

Changing the Default Action

With Red Hat Enterprise Linux 6.0, up to, and including version 6.2, the default action when **kdump** fails to create a core dump, the root file system is mounted and **/sbin/init** is run.

From Red Hat Enterprise Linux 6.3 onwards, the default behavior is to reboot the machine. This change was necessary to ensure that **kdump** could operate reliably using less reserved memory. To allow the previous behavior, the **mount_root_run_init** option has been added to [Table 32.3, "Supported actions"](#).

To change the default behavior, as **root**, open the **/etc/kdump.conf** configuration file in a text editor, remove the hash sign ("#") from the beginning of the **#default shell** line, and replace the value with a desired action as described in [Table 32.3, "Supported actions"](#).

Table 32.3. Supported actions

Option	Description
reboot	Reboot the system, losing the core in the process.
halt	Halt the system.
poweroff	Power off the system.
shell	Run the msh session from within the initramfs, allowing a user to record the core manually.
mount_root_run_init	Enable the default fallback behavior from Red Hat Enterprise Linux 6.2 and earlier.

For example:

```
default halt
```

Enabling the Service

To start the **kdump** daemon at boot time, type the following at a shell prompt as **root**:

```
chkconfig kdump on
```

This will enable the service for runlevels **2, 3, 4**, and **5**. Similarly, typing **chkconfig kdump off** will disable it for all runlevels. To start the service in the current session, use the following command as **root**:

```
service kdump start
```

For more information on runlevels and configuring services in general, see [Chapter 12, Services and Daemons](#).

32.2.4. Testing the Configuration



WARNING

The commands below will cause the kernel to crash. Use caution when following these steps, and by no means use them on a production machine.

To test the configuration, reboot the system with **kdump** enabled, and make sure that the service is running (see [Section 12.3, “Running Services”](#) for more information on how to run a service in Red Hat Enterprise Linux):

```
~]# service kdump status  
Kdump is operational
```

Then type the following commands at a shell prompt:

```
echo 1 > /proc/sys/kernel/sysrq  
echo c > /proc/sysrq-trigger
```

This will force the Linux kernel to crash, and the **address-YYYY-MM-DD-HH:MM:SS/vmcore** file will be copied to the location you have selected in the configuration (that is, to **/var/crash** by default).

32.3. ANALYZING THE CORE DUMP

To determine the cause of the system crash, you can use the **crash** utility, which provides an interactive prompt very similar to the GNU Debugger (GDB). This utility allows you to interactively analyze a running Linux system as well as a core dump created by **netdump**, **diskdump**, **xendump**, or **kdump**.



IMPORTANT

To analyze the **vmcore** dump file, you must have the crash and kernel-debuginfo packages installed. To install the crash package in your system, type the following at a shell prompt as **root**:

```
yum install crash
```

To install the kernel-debuginfo package, make sure that you have the `yum-utils` package installed and run the following command as **root**:

```
debuginfo-install kernel
```

Note that in order to use this command, you need to have access to the repository with debugging packages. If your system is registered with Red Hat Subscription Management, enable the **rhel-6-variant-debug-rpms** repository as described in [Section 8.4.4, "Viewing the Current Configuration"](#). If your system is registered with RHN Classic, subscribe the system to the **rhel-architecture-variant-6-debuginfo** channel as documented here: <https://access.redhat.com/site/solutions/9907>.

32.3.1. Running the crash Utility

To start the utility, type the command in the following form at a shell prompt:

```
crash /usr/lib/debug/lib/modules/kernel/vmlinux /var/crash/timestamp/vmcore
```

Note that the `kernel` version should be the same that was captured by **kdump**. To find out which kernel you are currently running, use the **uname -r** command.

Example 32.2. Running the crash utility

```
~]# crash /usr/lib/debug/lib/modules/2.6.32-69.el6.i686/vmlinux \
/var/crash/127.0.0.1-2010-08-25-08:45:02/vmcore

crash 5.0.0-23.el6
Copyright (C) 2002-2010 Red Hat, Inc.
Copyright (C) 2004, 2005, 2006 IBM Corporation
Copyright (C) 1999-2006 Hewlett-Packard Co
Copyright (C) 2005, 2006 Fujitsu Limited
Copyright (C) 2006, 2007 VA Linux Systems Japan K.K.
Copyright (C) 2005 NEC Corporation
Copyright (C) 1999, 2002, 2007 Silicon Graphics, Inc.
Copyright (C) 1999, 2000, 2001, 2002 Mission Critical Linux, Inc.
This program is free software, covered by the GNU General Public License,
and you are welcome to change it and/or redistribute copies of it under
certain conditions. Enter "help copying" to see the conditions.
This program has absolutely no warranty. Enter "help warranty" for details.
```

```
GNU gdb (GDB) 7.0
Copyright (C) 2009 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
```

This GDB was configured as "i686-pc-linux-gnu"...

```

KERNEL: /usr/lib/debug/lib/modules/2.6.32-69.el6.i686/vmlinux
DUMPFILE: /var/crash/127.0.0.1-2010-08-25-08:45:02/vmcore [PARTIAL DUMP]
CPUS: 4
DATE: Wed Aug 25 08:44:47 2010
UPTIME: 00:09:02
LOAD AVERAGE: 0.00, 0.01, 0.00
TASKS: 140
NODENAME: hp-dl320g5-02.lab.bos.redhat.com
RELEASE: 2.6.32-69.el6.i686
VERSION: #1 SMP Tue Aug 24 10:31:45 EDT 2010
MACHINE: i686 (2394 Mhz)
MEMORY: 8 GB
PANIC: "Oops: 0002 [#1] SMP " (check log for details)
PID: 5591
COMMAND: "bash"
TASK: f196d560 [THREAD_INFO: ef4da000]
CPU: 2
STATE: TASK_RUNNING (PANIC)
```

crash>

32.3.2. Displaying the Message Buffer

To display the kernel message buffer, type the **log** command at the interactive prompt.

Example 32.3. Displaying the kernel message buffer

```

crash> log
... several lines omitted ...
EIP: 0060:[<c068124f>] EFLAGS: 00010096 CPU: 2
EIP is at sysrq_handle_crash+0xf/0x20
EAX: 00000063 EBX: 00000063 ECX: c09e1c8c EDX: 00000000
ESI: c0a09ca0 EDI: 00000286 EBP: 00000000 ESP: ef4dbf24
DS: 007b ES: 007b FS: 00d8 GS: 00e0 SS: 0068
Process bash (pid: 5591, ti=ef4da000 task=f196d560 task.ti=ef4da000)
Stack:
c068146b c0960891 c0968653 00000003 00000000 00000002 efade5c0 c06814d0
<0> ffffffb c068150f b7776000 f2600c40 c0569ec4 ef4dbf9c 00000002 b7776000
<0> efade5c0 00000002 b7776000 c0569e60 c051de50 ef4dbf9c f196d560 ef4dbfb4
Call Trace:
[<c068146b>] ? __handle_sysrq+0xfb/0x160
[<c06814d0>] ? write_sysrq_trigger+0x0/0x50
[<c068150f>] ? write_sysrq_trigger+0x3f/0x50
[<c0569ec4>] ? proc_reg_write+0x64/0xa0
[<c0569e60>] ? proc_reg_write+0x0/0xa0
[<c051de50>] ? vfs_write+0xa0/0x190
[<c051e8d1>] ? sys_write+0x41/0x70
[<c0409adc>] ? syscall_call+0x7/0xb
Code: a0 c0 01 0f b6 41 03 19 d2 f7 d2 83 e2 03 83 e0 cf c1 e2 04 09 d0 88 41 03 f3 c3 90 c7 05
c8 1b 9e c0 01 00 00 00 0f ae f8 89 f6 <c6> 05 00 00 00 00 01 c3 89 f6 8d bc 27 00 00 00 00 8d 50
```

```
d0 83
EIP: [<c068124f>] sysrq_handle_crash+0xf/0x20 SS:ESP 0068:ef4dbf24
CR2: 0000000000000000
```

Type **help log** for more information on the command usage.



NOTE

The kernel message buffer includes the most essential information about the system crash and, as such, it is always dumped first in to the **vmcore-dmesg.txt** file. This is useful when an attempt to get the full **vmcore** file failed, for example because of lack of space on the target location. By default, **vmcore-dmesg.txt** is located in the **/var/crash/** directory.

32.3.3. Displaying a Backtrace

To display the kernel stack trace, type the **bt** command at the interactive prompt. You can use **bt pid** to display the backtrace of the selected process.

Example 32.4. Displaying the kernel stack trace

```
crash> bt
PID: 5591 TASK: f196d560 CPU: 2 COMMAND: "bash"
#0 [ef4dbdcc] crash_kexec at c0494922
#1 [ef4dbe20] oops_end at c080e402
#2 [ef4dbe34] no_context at c043089d
#3 [ef4dbe58] bad_area at c0430b26
#4 [ef4dbe6c] do_page_fault at c080fb9b
#5 [ef4bee4] error_code (via page_fault) at c080d809
    EAX: 00000063 EBX: 00000063 ECX: c09e1c8c EDX: 00000000 EBP: 00000000
    DS: 007b   ESI: c0a09ca0 ES: 007b   EDI: 00000286 GS: 00e0
    CS: 0060   EIP: c068124f ERR: ffffffff EFLAGS: 00010096
#6 [ef4dbf18] sysrq_handle_crash at c068124f
#7 [ef4dbf24] __handle_sysrq at c0681469
#8 [ef4dbf48] write_sysrq_trigger at c068150a
#9 [ef4dbf54] proc_reg_write at c0569ec2
#10 [ef4dbf74] vfs_write at c051de4e
#11 [ef4dbf94] sys_write at c051e8cc
#12 [ef4dbfb0] system_call at c0409ad5
    EAX: fffffda EBX: 00000001 ECX: b7776000 EDX: 00000002
    DS: 007b   ESI: 00000002 ES: 007b   EDI: b7776000
    SS: 007b   ESP: bfcb2088 EBP: bfcb20b4 GS: 0033
    CS: 0073   EIP: 00edc416 ERR: 00000004 EFLAGS: 00000246
```

Type **help bt** for more information on the command usage.

32.3.4. Displaying a Process Status

To display status of processes in the system, type the **ps** command at the interactive prompt. You can use **ps pid** to display the status of the selected process.

Example 32.5. Displaying status of processes in the system

```
crash> ps
  PID  PPID CPU TASK ST %MEM   VSZ RSS COMM
> 0    0 0 c09dc560 RU 0.0    0 0 [swapper]
> 0    0 1 f7072030 RU 0.0    0 0 [swapper]
  0    0 2 f70a3a90 RU 0.0    0 0 [swapper]
> 0    0 3 f70ac560 RU 0.0    0 0 [swapper]
  1    0 1 f705ba90 IN 0.0 2828 1424 init
... several lines omitted ...
  5566  1 1 f2592560 IN 0.0 12876 784 auditd
  5567  1 2 ef427560 IN 0.0 12876 784 auditd
  5587 5132 0 f196d030 IN 0.0 11064 3184 sshd
> 5591 5587 2 f196d560 RU 0.0 5084 1648 bash
```

Type **help ps** for more information on the command usage.

32.3.5. Displaying Virtual Memory Information

To display basic virtual memory information, type the **vm** command at the interactive prompt. You can use **vm pid** to display information on the selected process.

Example 32.6. Displaying virtual memory information of the current context

```
crash> vm
PID: 5591 TASK: f196d560 CPU: 2 COMMAND: "bash"
 MM   PGD   RSS TOTAL_VM
f19b5900 ef9c6000 1648k 5084k
 VMA   START   END FLAGS FILE
f1bb0310 242000 260000 8000875 /lib/ld-2.12.so
f26af0b8 260000 261000 8100871 /lib/ld-2.12.so
efbc275c 261000 262000 8100873 /lib/ld-2.12.so
efbc2a18 268000 3ed000 8000075 /lib/libc-2.12.so
efbc23d8 3ed000 3ee000 8000070 /lib/libc-2.12.so
efbc2888 3ee000 3f0000 8100071 /lib/libc-2.12.so
efbc2cd4 3f0000 3f1000 8100073 /lib/libc-2.12.so
efbc243c 3f1000 3f4000 100073
efbc28ec 3f6000 3f9000 8000075 /lib/libdl-2.12.so
efbc2568 3f9000 3fa000 8100071 /lib/libdl-2.12.so
efbc2f2c 3fa000 3fb000 8100073 /lib/libdl-2.12.so
f26af888 7e6000 7fc000 8000075 /lib/libtinfo.so.5.7
f26aff2c 7fc000 7ff000 8100073 /lib/libtinfo.so.5.7
efbc211c d83000 d8f000 8000075 /lib/libnss_files-2.12.so
efbc2504 d8f000 d90000 8100071 /lib/libnss_files-2.12.so
efbc2950 d90000 d91000 8100073 /lib/libnss_files-2.12.so
f26afe00 edc000 edd000 4040075
f1bb0a18 8047000 8118000 8001875 /bin/bash
f1bb01e4 8118000 811d000 8101873 /bin/bash
f1bb0c70 811d000 8122000 100073
f26afae0 9fd9000 9ffa000 100073
... several lines omitted ...
```

Type **help vm** for more information on the command usage.

32.3.6. Displaying Open Files

To display information about open files, type the **files** command at the interactive prompt. You can use **files pid** to display files opened by the selected process.

Example 32.7. Displaying information about open files of the current context

```
crash> files
PID: 5591 TASK: f196d560 CPU: 2 COMMAND: "bash"
ROOT: / CWD: /root
FD FILE DENTRY INODE TYPE PATH
0 f734f640 eedc2c6c eecd6048 CHR /pts/0
1 efade5c0 eee14090 f00431d4 REG /proc/sysrq-trigger
2 f734f640 eedc2c6c eecd6048 CHR /pts/0
10 f734f640 eedc2c6c eecd6048 CHR /pts/0
255 f734f640 eedc2c6c eecd6048 CHR /pts/0
```

Type **help files** for more information on the command usage.

32.3.7. Exiting the Utility

To exit the interactive prompt and terminate **crash**, type **exit** or **q**.

Example 32.8. Exiting the crash utility

```
crash> exit
~]#
```

32.4. USING FADUMP ON IBM POWERPC HARDWARE

Starting with Red Hat Enterprise Linux 6.8 an alternative dumping mechanism to **kdump**, the **firmware-assisted dump (fadump)**, is available. The **fadump** feature is supported only on IBM Power Systems. The goal of **fadump** is to enable the dump of a crashed system, and to do so from a fully-reset system, and to minimize the total elapsed time until the system is back in production use. The **fadump** feature is integrated with **kdump** infrastructure present in the user space to seemlessly switch between **kdump** and **fadump** mechanisms.

Firmware-assisted dump (**fadump**) is a reliable alternative to **kexec-kdump** available on IBM PowerPC LPARS. It captures vmcore from a fully-reset system with PCI and I/O devices reinitialized. While this mechanism uses the firmware to preserve the memory in case of a crash, it reuses the **kdump** userspace scripts to save the vmcore"

To achieve this, **fadump** registers the regions of memory that must be preserved in the event of a crash with the system firmware. These regions consist of all the system memory contents, except the boot memory, system registers and hardware Page Table Entries (PTEs).



NOTE

The area of memory not preserved and known as **boot memory** is the amount of RAM required to successfully boot the kernel after a crash event. By default, the boot memory size is 256MB or 5% of total system RAM, whichever is larger.

Unlike a **kexec**-initiated event, the **fadump** process uses the production kernel to recover a crash dump. When booting after a crash, PowerPC hardware makes the device node **/proc/device-tree/rtas/ibm,kernel-dump** available to **procfs**, which the fadump-aware **kdump** scripts check for to save the vmcore. After this has completed, the system is rebooted cleanly.

Enabling fadump

1. Install and configure **kdump** as described in [Section 32.1, “Installing the kdump Service”](#) and [Section 32.2, “Configuring the kdump Service”](#).
2. Add **fadump=on** to the **GRUB_CMDLINE_LINUX** line in **/etc/default/grub**:

```
GRUB_CMDLINE_LINUX="rd.lvm.lv=rhel/swap crashkernel=auto rd.lvm.lv=rhel/root rhgb quiet fadump=on"
```

3. (optional) If you want to specify reserved boot memory instead of accepting the defaults, add **fadump_reserve_mem=xxM** to **GRUB_CMDLINE_LINUX** in **/etc/default/grub**, where **xx** is the amount of the memory required in megabytes:

```
GRUB_CMDLINE_LINUX="rd.lvm.lv=rhel/swap crashkernel=auto rd.lvm.lv=rhel/root rhgb quiet fadump=on fadump_reserve_mem=xxM"
```



IMPORTANT

As with all boot configuration options, it is strongly recommended that you test the configuration before it is needed. If you observe Out of Memory (OOM) errors when booting from the crash kernel, increase the value specified in **fadump_reserve_mem=** until the crash kernel can boot cleanly. Some trial and error may be required in this case.

32.5. USING SADUMP ON FUJITSU PRIMEQUEST SYSTEMS

On Fujitsu PRIMEQUEST systems, you can also enable the stand-alone dump (**sadump**) functionality provided by the hardware. This utility operates by combining **kdump**, which is a standard part of Red Hat Enterprise Linux, and the additional **sadump** BIOS-based function provided by Fujitsu.



NOTE

For the purpose of ensuring a dump is captured in the event of an unexpected reboot, Fujitsu recommend that **sadump** is always enabled on PRIMEQUEST hardware.

The **sadump** utility is usually invoked when a **kdump** cannot be processed because Red Hat Enterprise Linux has became unresponsive. These conditions can include:

- Red Hat Enterprise Linux panic or hang before **kdump** starts
- An error while **kdump** is working

How to use sadump

To use **sadump**, complete the following steps:

1. Install the following packages according to the kernel version in use:

```
# yum install kernel-debuginfo kernel-debuginfo-common
```

2. Configure UEFI for **sadump**

For more details, see the FUJITSU Server PRIMEQUEST 2000 Series Installation Manual.

3. Configure Red Hat Enterprise Linux for **sadump**

For more details, see [Section 32.5.1, “Configure Red Hat Enterprise Linux for sadump”](#).

4. Start **sadump**

For more details, see the FUJITSU Server PRIMEQUEST 2000 Series Installation Manual.

5. Check the memory dump

For more details, see [Section 32.5.2, “Check the memory dump”](#).

32.5.1. Configure Red Hat Enterprise Linux for sadump

1. Install and configure **kdump** as described in [Section 32.1, “Installing the kdump Service”](#) and [Section 32.2, “Configuring the kdump Service”](#).
2. Ensure that **kdump** starts as expected for **sadump**:

- a. Configure Red Hat Enterprise Linux to not reboot after a kernel panic:

By default, Red Hat Enterprise Linux reboots automatically after a kernel panic, which prevents **sadump** to start. To avoid this behavior, configure the **/etc/sysctl.conf** file as follows:

```
kernel.panic=0
```

- b. Configure Red Hat Enterprise Linux to start **kdump** by Nonmaskable Interrupt (NMI):

In the procedure of starting **sadump**, starting **kdump** by NMI is needed at first.

Configure **/etc/sysctl.conf** as follows:

```
kernel.unknown_nmi_panic=1
```

3. Ensure that **kdump** behaves correctly for **sadump**:

- a. Configure Red Hat Enterprise Linux to stop after **kdump**:

By default, Red Hat Enterprise Linux reboots automatically when **kdump** fails, which prevents **sadump** to start. To avoid this behavior, configure the **/etc/kdump.conf** file as follows:

```
default halt
```

or

default shell

- b. Configure Red Hat Enterprise Linux to start **sadump**:

Configure **/etc/kdump.conf** to not block the System Management Interrupt (SMI) and thus to enable **sadump** to start:

blacklist kvm-intel

32.5.2. Check the memory dump

When the memory dump is finished and Red Hat Enterprise Linux is rebooted, you can check the memory dump with the **crash** command by opening the special device file.

The following example shows how to check the memory dump which is saved on /dev/sdb1.

Example 32.9. Checking Dump Integrity

```
crash /usr/lib/debug/lib/modules/2.6.32-358.el6.x86_64/vmlinux /dev/sdb1
```

32.6. ADDITIONAL RESOURCES

Installed Documentation

- **kdump.conf(5)** – a manual page for the **/etc/kdump.conf** configuration file containing the full documentation of available options.
- **makedumpfile(8)** – a manual page for the **makedumpfile** core collector.
- **kexec(8)** – a manual page for **kexec**.
- **crash(8)** – a manual page for the **crash** utility.
- **/usr/share/doc/kexec-tools-version/kexec-kdump-howto.txt** – an overview of the **kdump** and **kexec** installation and usage.
- **/usr/share/doc/kexec-tools-version/fadump-howto.txt** – more details about the **fadump** mechanism, including PowerPC-specific methods of resetting hardware. Replace *version* with kexec-tools version installed on your system.

Useful Websites

<https://access.redhat.com/kb/docs/DOC-6039>

The Red Hat Knowledgebase article about the **kexec** and **kdump** configuration.

<https://access.redhat.com/kb/docs/DOC-45183>

The Red Hat Knowledgebase article about supported **kdump** targets.

<http://people.redhat.com/anderson/>

The **crash** utility homepage.

PART IX. SYSTEM RECOVERY

This part introduces system recovery modes on Red Hat Enterprise Linux 6 and advises users on how to repair the system in certain situations. It also describes how to use the Relax-and-Recover (ReaR) disaster recovery and system migration utility.

CHAPTER 33. SYSTEM RECOVERY

Red Hat Enterprise Linux 6 offers three system recovery modes, **rescue mode**, **single-user mode**, and **emergency mode** that can be used to repair malfunctioning systems. This chapter describes how to boot into each system recovery mode and gives guidance to resolve certain problems that can only be solved with help of system recovery modes.

These are the usual reasons why you may need to boot to one of the system recovery modes:

- You are unable to boot normally into Red Hat Enterprise Linux (runlevel 3 or 5).
- You need to resolve hardware or software problems that cannot be resolved while the system is running normally, or you want to access some important files off of your hard drive.
- You forgot the root password.

Some of the problems behind are further discussed in [Section 33.4, "Resolving Problems in System Recovery Modes"](#).

33.1. RESCUE MODE

Rescue mode provides the ability to boot a small Red Hat Enterprise Linux environment entirely from external media, such as CD-ROM or USB drive, instead of the system's hard drive. It contains command-line utilities for repairing a wide variety of issues. In this mode, you can mount file systems as read-only or even to not mount them at all, blacklist or add drivers provided on a driver disc, install or upgrade system packages, or manage partitions.

To boot into **rescue mode** follow this procedure:

Procedure 33.1. Booting into Rescue Mode

1. Boot the system from either minimal boot media, or a full installation DVD or USB drive, and wait for the boot menu to appear. For details about booting the system from the chosen media, see the respective chapters in the [Installation Guide](#).
2. From the boot menu, append the **rescue** keyword as a kernel parameter to the boot command line.
3. If your system requires a third-party driver provided on a *driver disc* to boot, append the additional option **dd** to the boot command line to load that driver:

```
rescue dd
```

For more information about using a disc driver at boot time, see the respective chapters in the [Installation Guide](#).

4. If a driver that is a part of the Red Hat Enterprise Linux 6 distribution prevents the system from booting, blacklist that driver by appending the **rdblacklist** option to the boot command line:

```
rescue rdblacklist=driver_name
```
5. Answer a few basic questions and select the location of a valid rescue image as you are prompted to. Select the relevant type from **Local CD-ROM**, **Hard Drive**, **NFS image**, **FTP**, or **HTTP**. The selected location must contain a valid installation tree, and the installation tree must

be for the same version of Red Hat Enterprise Linux as is the disk from which you booted. For more information about how to setup an installation tree on a hard drive, NFS server, FTP server, or HTTP server, see the respective chapters in the [Installation Guide](#).

If you select a rescue image that does not require a network connection, you are asked whether or not you want to establish a network connection. A network connection is useful if you need to backup files to a different computer or install some RPM packages from a shared network location.

6. The following message is displayed:

The rescue environment will now attempt to find your Linux installation and mount it under the directory `/mnt/sysimage`. You can then make any changes required to your system. If you want to proceed with this step choose 'Continue'. You can also choose to mount your file systems read-only instead of read-write by choosing 'Read-only'. If for some reason this process fails you can choose 'Skip' and this step will be skipped and you will go directly to a command shell.

If you select **Continue**, the system attempts to mount your root partition under the `/mnt/sysimage/` directory. The root partition typically contains several file systems, such as `/home/`, `/boot/`, and `/var/`, which are automatically mounted to the correct locations. If mounting the partition fails, you will be notified. If you select **Read-Only**, the system attempts to mount your file systems under the directory `/mnt/sysimage/`, but in read-only mode. If you select **Skip**, your file systems will not be mounted. Choose **Skip** if you think your file system is corrupted.

7. Once you have your system in rescue mode, the following prompt appears on the virtual console (VC) 1 and VC 2. Use the **Ctrl-Alt-F1** key combination to access VC 1 and **Ctrl-Alt-F2** to access VC 2:

```
sh-3.00b#
```

If you selected **Continue** to mount your partitions automatically and they were mounted successfully, you are in **single-user mode**.

Even if your file system is mounted, the default root partition while in **rescue mode** is a temporary root partition, not the root partition of the file system used during normal user mode (runlevel 3 or 5). If you selected to mount your file system and it mounted successfully, you can change the root partition of the rescue mode environment to the root partition of your file system by executing the following command:

```
sh-3.00b# chroot /mnt/sysimage
```

This is useful if you need to run commands, such as **rpm**, that require your root partition to be mounted as `/`. To exit the **chroot** environment, type **exit** to return to the prompt.

If you selected **Skip**, you can still try to mount a partition or a LVM2 logical volume manually inside **rescue mode** by creating a directory and typing the following command:

```
sh-3.00b# mkdir /directory
sh-3.00b# mount -t ext4 /dev/mapper/VolGroup00-LogVol02 /directory
```

where `/directory` is a directory that you have created and `/dev/mapper/VolGroup00-LogVol02` is the LVM2 logical volume you want to mount. If the partition is of **ext2** or **ext3** type, replace **ext4** with **ext2** or **ext3** respectively.

If you do not know the names of all physical partitions, use the following command to list them:

```
sh-3.00b# fdisk -l
```

If you do not know the names of all LVM2 physical volumes, volume groups, or logical volumes, use the **pvdisplay**, **vgdisplay** or **lvdisplay** commands, respectively.

From the prompt, you can run many useful commands, such as:

- **ssh**, **scp**, and **ping** if the network is started
- **dump** and **restore** for users with tape drives
- **parted** and **fdisk** for managing partitions
- **rpm** for installing or upgrading software
- **vi** for editing text files

33.2. SINGLE-USER MODE

Single-user mode provides a Linux environment for a single user that allows you to recover your system from problems that cannot be resolved in networked multi-user environment. You do not need an external boot device to be able to boot into **single-user mode**, and you can switch into it directly while the system is running. To switch into **single-user mode** on the running system, issue the following command from the command line:

```
~]# init 1
```

In **single-user mode**, the system boots with your local file systems mounted, many important services running, and a usable maintenance shell that allows you to perform many of the usual system commands. Therefore, **single-user mode** is mostly useful for resolving problems when the system boots but does not function properly or you cannot log into it.



WARNING

The **single-user mode** automatically tries to mount your local file systems. Booting to **single-user mode** could result in loss of data if any of your local file systems cannot be successfully mounted.

To boot into **single-user mode** follow this procedure:

Procedure 33.2. Booting into Single-User Mode

1. At the GRUB boot screen, press any key to enter the GRUB interactive menu.
2. Select **Red Hat Enterprise Linux** with the version of the kernel that you want to boot and press the **a** to append the line.
3. Type **single** as a separate word at the end of the line and press **Enter** to exit GRUB edit mode. Alternatively, you can type **1** instead of single.

33.3. EMERGENCY MODE

Emergency mode, provides the minimal bootable environment and allows you to repair your system even in situations when **rescue mode** is unavailable. In **emergency mode**, the system mounts only the **root** file system, and it is mounted as read-only. Also, the system does not activate any network interfaces and only a minimum of the essential services are set up. The system does not load any **init** scripts, therefore you can still mount file systems to recover data that would be lost during a re-installation if **init** is corrupted or not working.

To boot into **emergency mode** follow this procedure:

Procedure 33.3. Booting into Emergency Mode

1. At the GRUB boot screen, press any key to enter the GRUB interactive menu.
2. Select **Red Hat Enterprise Linux** with the version of the kernel that you want to boot and press the **a** to append the line.
3. Type **emergency** as a separate word at the end of the line and press **Enter** to exit GRUB edit mode.

33.4. RESOLVING PROBLEMS IN SYSTEM RECOVERY MODES

This section provides several procedures that explain how to resolve some of the most common problems that needs to be addressed in some of the system recovery modes.

The following procedure shows how to reset a **root** password:

Procedure 33.4. Resetting a Root Password

1. Boot to **single-user mode** as described in [Procedure 33.2, “Booting into Single-User Mode”](#).
2. Run the **passwd** command from the maintenance shell command line.

One of the most common causes for an unbootable system is overwriting of the Master Boot Record (MBR) that originally contained the GRUB boot loader. If the boot loader is overwritten, you cannot boot Red Hat Enterprise Linux unless you reconfigure the boot loader in **rescue mode**.

To reinstall GRUB on the MBR of your hard drive, proceed with the following procedure:

Procedure 33.5. Reinstalling the GRUB Boot Loader

1. Boot to **rescue mode** as described in [Procedure 33.1, “Booting into Rescue Mode”](#). Ensure that you mount the system's root partition in read-write mode.
2. Execute the following command to change the root partition:

```
sh-3.00b# chroot /mnt/sysimage
```

3. Run the following command to reinstall the GRUB boot loader:

```
sh-3.00b# /sbin/grub-install boot_part
```

where *boot_part* is your boot partition (typically, **/dev/sda**).

4. Review the **/boot/grub/grub.conf** file, as additional entries may be needed for GRUB to control additional operating systems.
5. Reboot the system.

Another common problem that would render your system unbootable is a change of your root partition number. This can usually happen when resizing a partition or creating a new partition after installation. If the partition number of your root partition changes, the GRUB boot loader might not be able to find it to mount the partition. To fix this problem, boot into **rescue mode** and modify the **/boot/grub/grub.conf** file.

A malfunctioning or missing driver can prevent a system from booting normally. You can use the **RPM** package manager to remove malfunctioning drivers or to add updated or missing drivers in **rescue mode**. If you cannot remove a malfunctioning driver for some reason, you can instead blacklist the driver so that it does not load at boot time.



NOTE

When you install a driver from a driver disc, the driver disc updates all initramfs images on the system to use this driver. If a problem with a driver prevents a system from booting, you cannot rely on booting the system from another initramfs image.

To remove a malfunctioning driver that prevents the system from booting, follow this procedure:

Procedure 33.6. Remove a Driver in Rescue Mode

1. Boot to **rescue mode** as described in [Procedure 33.1, “Booting into Rescue Mode”](#). Ensure that you mount the system's root partition in read-write mode.
2. Change the root directory to **/mnt/sysimage/**:

```
sh-3.00b# chroot /mnt/sysimage
```

3. Run the following command to remove the driver package:

```
sh-3.00b# rpm -e driver_name
```

4. Exit the **chroot** environment:

```
sh-3.00b# exit
```

5. Reboot the system.

To install a missing driver that prevents the system from booting, follow this procedure:

Procedure 33.7. Installing a Driver in Rescue Mode

1. Boot to **rescue mode** as described in [Procedure 33.1, “Booting into Rescue Mode”](#). Ensure that you mount the system's root partition in read-write mode.
2. Mount a media with an RPM package that contains the driver and copy the package to a location of your choice under the **/mnt/sysimage/** directory, for example: **/mnt/sysimage/root/drivers/**.

3. Change the root directory to **/mnt/sysimage/**:

```
sh-3.00b# chroot /mnt/sysimage
```

4. Run the following command to install the driver package:

```
sh-3.00b# rpm -ihv /root/drivers/package_name
```

Note that **/root/drivers** in this chroot environment is **/mnt/sysimage/root/drivers** in the original rescue environment.

5. Exit the **chroot** environment:

```
sh-3.00b# exit
```

6. Reboot the system.

To blacklist a driver that prevents the system from booting and to ensure that this driver cannot be loaded after the root device is mounted, follow this procedure:

Procedure 33.8. Blacklisting a Driver in Rescue Mode

1. Boot to **rescue mode** with the command **linux rescue rdblacklist=driver_name**, where **driver_name** is the driver that you need to blacklist. Follow the instructions in [Procedure 33.1, “Booting into Rescue Mode”](#) and ensure that you mount the system's root partition in read-write mode.

2. Open the **/boot/grub/grub.conf** file in the **vi** editor:

```
sh-3.00b# vi /boot/grub/grub.conf
```

3. Identify the default kernel used to boot the system. Each kernel is specified in the **grub.conf** file with a group of lines that begins **title**. The default kernel is specified by the **default** parameter near the start of the file. A value of **0** refers to the kernel described in the first group of lines, a value of **1** refers to the kernel described in the second group, and higher values refer to subsequent kernels in turn.

4. Edit the **kernel** line of the group to include the option **rdblacklist=driver_name**, where **driver_name** is the driver that you need to blacklist. For example:

```
kernel /vmlinuz-2.6.32-71.18-2.el6.i686 ro root=/dev/sda1 rhgb quiet rdblacklist=driver_name
```

5. Save the file and exit the **vi** editor by typing:

```
:wq
```

6. Run the following command to create a new file **/etc/modprobe.d/driver_name.conf** that will ensure blacklisting of the driver after the root partition is mounted:

```
echo "install driver_name" > /mnt/sysimage/etc/modprobe.d/driver_name.conf
```

7. Reboot the system.

CHAPTER 34. RELAX-AND-RECOVER (REAR)

When a software or hardware failure breaks the system, the system administrator faces three tasks to restore it to the fully functioning state on a new hardware environment:

1. booting a rescue system on the new hardware
2. replicating the original storage layout
3. restoring user and system files

Most backup software solves only the third problem. To solve the first and second problems, use *Relax-and-Recover (ReaR)*, a disaster recovery and system migration utility.

Backup software creates backups. ReaR complements backup software by creating a *rescue system*. Booting the rescue system on a new hardware allows you to issue the **rear recover** command, which starts the recovery process. During this process, ReaR replicates the partition layout and filesystems, prompts for restoring user and system files from the backup created by backup software, and finally installs the boot loader. By default, the rescue system created by ReaR only restores the storage layout and the boot loader, but not the actual user and system files.

This chapter describes how to use ReaR.

34.1. BASIC REAR USAGE

34.1.1. Installing ReaR

Install the rear package by running the following command as root:

```
~]# yum install rear
```

34.1.2. Configuring ReaR

ReaR is configured in the **/etc/rear/local.conf** file. Specify the rescue system configuration by adding these lines:

```
OUTPUT=output format
OUTPUT_URL=output location
```

Substitute *output format* with rescue system format, for example, **ISO** for an ISO disk image or **USB** for a bootable USB.

Substitute *output location* with where it will be put, for example, **file:///mnt/rescue_system/** for a local filesystem directory or **sftp://backup:password@192.168.0.0/** for an SFTP directory.

Example 34.1. Configuring Rescue System Format and Location

To configure ReaR to output the rescue system as an ISO image into the **/mnt/rescue_system/** directory, add these lines to the **/etc/rear/local.conf** file:

```
OUTPUT=ISO
OUTPUT_URL=file:///mnt/rescue_system/
```

See section "Rescue Image Configuration" of the rear(8) man page for a list of all options.

34.1.3. Creating a Rescue System

The following example shows how to create a rescue system with verbose output:

```
~]# rear -v mkrescue
Relax-and-Recover 1.17.2 / Git
Using log file: /var/log/rear/rear-rhel68.log
mkdir: created directory '/var/lib/rear/output'
Creating disk layout
Creating root filesystem layout
TIP: To login as root via ssh you need to set up /root/.ssh/authorized_keys or
SSH_ROOT_PASSWORD in your configuration file
Copying files and directories
Copying binaries and libraries
Copying kernel modules
Creating initramfs
Making ISO image
Wrote ISO image: /var/lib/rear/output/rear-rhel68.iso (82M)
Copying resulting files to file location
```

With the configuration from [Example 34.1, "Configuring Rescue System Format and Location"](#), ReaR prints the above output. The last two lines confirm that the rescue system has been successfully created and copied to the configured backup location **/mnt/rescue_system/**. Because the system's host name is **rhel-68**, the backup location now contains directory **rhel-68/** with the rescue system and auxiliary files:

```
~]# ls -lh /mnt/rescue_system/rhel68/
total 82M
-rw----- 1 root root 202 May  9 11:46 README
-rw----- 1 root root 160K May  9 11:46 rear.log
-rw----- 1 root root 82M May  9 11:46 rear-rhel68.iso
-rw----- 1 root root 275 May  9 11:46 VERSION
```

Transfer the rescue system to an external medium to not lose it in case of a disaster.

34.1.4. Scheduling ReaR

To schedule ReaR to regularly create a rescue system using the **cron** job scheduler, add the following line to the **/etc/crontab** file:

```
minute hour day_of_month month day_of_week root /usr/sbin/rear mkrescue
```

Substitute the above command with the cron time specification (described in detail in [Section 27.1.4, "Configuring Cron Jobs"](#)).

Example 34.2. Scheduling ReaR

To make ReaR create a rescue system at 22:00 every weekday, add this line to the **/etc/crontab** file:

```
0 22 * * 1-5 root /usr/sbin/rear mkrescue
```

34.1.5. Performing a System Rescue

To perform a restore or migration:

1. Boot the rescue system on the new hardware. For example, burn the ISO image to a DVD and boot from the DVD.
2. In the console interface, select the "Recover" option:

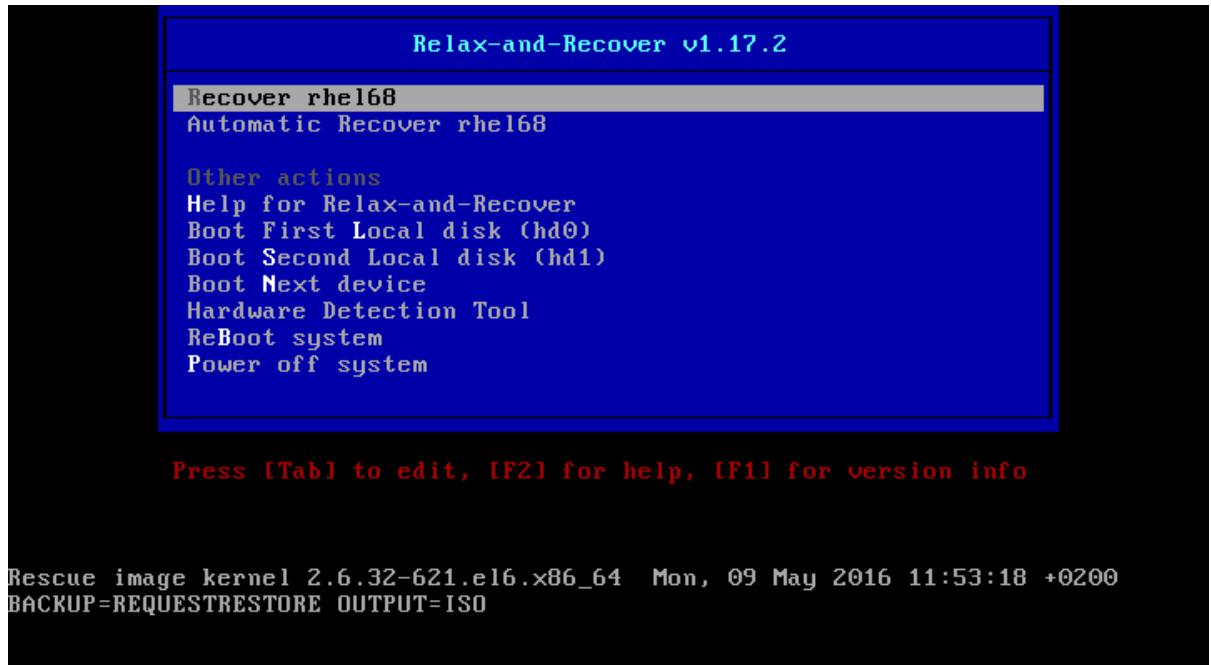


Figure 34.1. Rescue system: menu

3. You are taken to the prompt:

```

Attempting to start the DHCP client daemon
Running 60-network-devices.sh...
Running 62-routing.sh...
Running 99-makedev.sh...
* * * Rescue System is ready * * *

Relax-and-Recover 1.17.2 / Git

Relax-and-Recover comes with ABSOLUTELY NO WARRANTY; for details see
the GNU General Public License at: http://www.gnu.org/licenses/gpl.html

Host rhel68 using Backup REQUESTRESTORE and Output ISO
Build date: Mon, 09 May 2016 11:53:01 +0200

Red Hat Enterprise Linux Server release 6.8 (Santiago)
Kernel 2.6.32-621.el6.x86_64 on an x86_64

rhel68 login: root (automatic login)

Welcome to Relax and Recover. Run "rear recover" to restore your system !
RESCUE rhel68:~ #

```

Figure 34.2. Rescue system: prompt

**WARNING**

Once you have started recovery in the next step, it probably cannot be undone and you may lose anything stored on the physical disks of the system.

- Run the **rear recover** command to perform the restore or migration. The rescue system then recreates the partition layout and filesystems:

```
RESCUE rhel68:~ # rear recover
Relax-and-Recover 1.17.2 / Git
Using log file: /var/log/rear/rear-rhel68.log
NOTICE: Will do driver migration
Comparing disks.
Disk configuration is identical, proceeding with restore.
Start system layout restoration.
Creating partitions for disk /dev/sda (msdos)
Creating LVM PV /dev/sda2
Restoring LVM VG VolGroup
Sleeping 3 seconds to let udev or systemd-udevd create their devices...
Creating ext4-filesystem / on /dev/mapper/VolGroup-lv_root
Mounting filesystem /
Creating ext4-filesystem /boot on /dev/sda1
Mounting filesystem /boot
Creating swap on /dev/mapper/VolGroup-lv_swap
Disk layout created.
Please start the restore process on your backup host.

Make sure that you restore the data into '/mnt/local' instead of '/' because the
hard disks of the recovered system are mounted there.

Please restore your backup in the provided shell and, when finished, type exit
in the shell to continue recovery.
rear> _
```

Figure 34.3. Rescue system: running "rear recover"

- Restore user and system files from the backup into the **/mnt/local/** directory.

Example 34.3. Restoring User and System Files

In this example, the backup file is a **tar** archive created per instructions in [Section 34.2.1.1, "Configuring the Internal Backup Method"](#). First, copy the archive from its storage, then unpack the files into **/mnt/local/**, then delete the archive:

```
~]# scp root@192.168.122.6:/srv/backup/rhel68/backup.tar.gz /mnt/local/
~]# tar xf /mnt/local/backup.tar.gz -C /mnt/local/
~]# rm -f /mnt/local/backup.tar.gz
```

The new storage has to have enough space both for the archive and the extracted files.

- Verify that the files have been restored:

```
~]# ls /mnt/local/
```

```

Mounting filesystem /
Creating ext4-filesystem /boot on /dev/sda1
Mounting filesystem /boot
Creating swap on /dev/mapper/vg_rhel68-lv_swap
Disk layout created.
Please start the restore process on your backup host.

Make sure that you restore the data into '/mnt/local' instead of '/' because the
hard disks of the recovered system are mounted there.

Please restore your backup in the provided shell and, when finished, type exit
in the shell to continue recovery.
rear> scp -r root@192.168.122.6:/srv/backup/rhel68/backup.tar.gz /mnt/local/
The authenticity of host '192.168.122.6 (192.168.122.6)' can't be established.
RSA key fingerprint is c6:b0:9b:52:88:5a:c5:a1:3d:4c:40:ed:7a:88:33:77.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.122.6' (RSA) to the list of known hosts.
root@192.168.122.6's password:
backup.tar.gz          100%  736MB  18.0MB/s   00:41
rear> tar xf /mnt/local/backup.tar.gz -C /mnt/local/
rear> rm -f /mnt/local/backup.tar.gz
rear> ls /mnt/local
bin  cgroup  etc    lib    lost+found  misc  net  proc  sbin    srv  tmp  var
boot dev      home  lib64  media       mnt  opt  root  selinux sys  usr
rear> _

```

Figure 34.4. Rescue system: restoring user and system files from the backup

7. Ensure that SELinux relabels the files on the next boot:

```

~]# touch /mnt/local/.autorelabel

```

Otherwise you may be unable to log in the system, because the **/etc/passwd** file may have the incorrect SELinux context.

8. Finish the recovery and reboot the system:

```

rear> scp -r root@192.168.122.6:/srv/backup/rhel68/backup.tar.gz /mnt/local/
The authenticity of host '192.168.122.6 (192.168.122.6)' can't be established.
RSA key fingerprint is c6:b0:9b:52:88:5a:c5:a1:3d:4c:40:ed:7a:88:33:77.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.122.6' (RSA) to the list of known hosts.
root@192.168.122.6's password:
backup.tar.gz          100%  736MB  18.0MB/s   00:41
rear> tar xf /mnt/local/backup.tar.gz -C /mnt/local/
rear> rm -f /mnt/local/backup.tar.gz
rear> ls /mnt/local
bin  cgroup  etc    lib    lost+found  misc  net  proc  sbin    srv  tmp  var
boot dev      home  lib64  media       mnt  opt  root  selinux sys  usr
rear> touch /mnt/local/.autorelabel
rear> exit
Did you restore the backup to /mnt/local ? Are you ready to continue recovery ?
y
exit
Updated initramfs with new drivers for Kernel 2.6.32-642.el6.x86_64.
Installing GRUB boot loader
Updating udev configuration (70-persistent-net.rules)
Updating udev configuration (70-persistent-cd.rules)

Finished recovering your system. You can explore it under '/mnt/local'.

RESCUE rhel68:~ # reboot.

```

Figure 34.5. Rescue system: finishing recovery

ReaR will then reinstall the boot loader. Upon reboot, SELinux will relabel the whole filesystem.

Then you will be able to log in to the recovered system.

34.2. INTEGRATING REAR WITH BACKUP SOFTWARE

The main purpose of ReaR is to produce a rescue system, but it can also be integrated with backup software. What integration means is different for the built-in, supported, and unsupported backup methods.

34.2.1. The Built-in Backup Method

ReaR ships with a built-in, or internal, backup method. This method is fully integrated with ReaR, which has these advantages:

- a rescue system and a full-system backup can be created using a single **rear mkbackup** command
- the rescue system restores files from the backup automatically

As a result, ReaR can cover the whole process of creating both the rescue system and the full-system backup.

34.2.1.1. Configuring the Internal Backup Method

To make ReaR use its internal backup method, add these lines to **/etc/rear/local.conf**:

```
BACKUP=NETFS
BACKUP_URL=backup location
```

These lines configure ReaR to create an archive with a full-system backup using the **tar** command. Substitute *backup location* with one of the options from the "Backup Software Integration" section of the **rear(8)** man page. Make sure that the backup location has enough space.

Example 34.4. Adding tar Backups

To expand the example in [Section 34.1, "Basic ReaR Usage"](#), configure ReaR to also output a **tar** full-system backup into the **/srv/backup/** directory:

```
OUTPUT=ISO
OUTPUT_URL=file:///mnt/rescue_system/
BACKUP=NETFS
BACKUP_URL=file:///srv/backup/
```

The internal backup method allows further configuration.

- To keep old backup archives when new ones are created, add this line:


```
NETFS_KEEP_OLD_BACKUP_COPY=y
```
- By default, ReaR creates a full backup on each run. To make the backups incremental, meaning that only the changed files are backed up on each run, add this line:


```
BACKUP_TYPE=incremental
```

This automatically sets **NETFS_KEEP_OLD_BACKUP_COPY** to **y**.

- To ensure that a full backup is done regularly in addition to incremental backups, add this line:

```
FULLBACKUPDAY="Day"
```

Substitute "Day" with one of the "Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun".

- ReaR can also include both the rescue system and the backup in the ISO image. To achieve this, set the **BACKUP_URL** directive to **iso:///backup/**:

```
BACKUP_URL=iso:///backup/
```

This is the simplest method of full-system backup, because the rescue system does not need the user to fetch the backup during recovery. However, it needs more storage. Also, single-ISO backups cannot be incremental.



NOTE

Currently ReaR creates two copies of the ISO image, thus consuming two times more storage. For more information, see note *ReaR creates two ISO images instead of one* in [Red Hat Enterprise Linux 6 Release Notes](#).

Example 34.5. Configuring Single-ISO Rescue System and Backups

This configuration creates a rescue system and a backup file as a single ISO image and puts it into the **/srv/backup/** directory:

```
OUTPUT=ISO
OUTPUT_URL=file:///srv/backup/
BACKUP=NETFS
BACKUP_URL=iso:///backup/
```

- To use **rsync** instead of **tar**, add this line:

```
BACKUP_PROG=rsync
```

Note that incremental backups are only supported when using **tar**.

34.2.1.2. Creating a Backup Using the Internal Backup Method

With **BACKUP=NETFS** set, ReaR can create either a rescue system, a backup file, or both.

- To create a *rescue system only*, run:

```
rear mkrescue
```

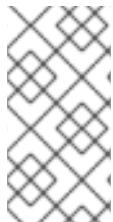
- To create a *backup only*, run:

```
rear mkbackuponly
```

- To create a *rescue system and a backup*, run:

rear mkbackup

Note that triggering backup with ReaR is only possible if using the NETFS method. ReaR cannot trigger other backup methods.



NOTE

When restoring, the rescue system created with the **BACKUP=NETFS** setting expects the backup to be present before executing **rear recover**. Hence, once the rescue system boots, copy the backup file into the directory specified in **BACKUP_URL**, unless using a single ISO image. Only then run **rear recover**.

To avoid recreating the rescue system unnecessarily, you can check whether storage layout has changed since the last rescue system was created using these commands:

```
~]# rear checklayout
~]# echo $?
```

Non-zero status indicates a change in disk layout. Non-zero status is also returned if ReaR configuration has changed.



IMPORTANT

The **rear checklayout** command does not check whether a rescue system is currently present in the output location, and can return 0 even if it is not there. So it does not guarantee that a rescue system is available, only that the layout has not changed since the last rescue system has been created.

Example 34.6. Using rear checklayout

To create a rescue system, but only if the layout has changed, use this command:

```
~]# rear checklayout || rear mkrescue
```

34.2.2. Supported Backup Methods

In addition to the NETFS internal backup method, ReaR supports several external backup methods. This means that the rescue system restores files from the backup automatically, but the backup creation cannot be triggered using ReaR.

For a list and configuration options of the supported external backup methods, see the "Backup Software Integration" section of the `rear(8)` man page.

34.2.3. Unsupported Backup Methods

With unsupported backup methods, there are two options:

1. The rescue system prompts the user to manually restore the files. This scenario is the one described in "Basic ReaR Usage", except for the backup file format, which may take a different form than a `tar` archive.

2. ReaR executes the custom commands provided by the user. To configure this, set the **BACKUP** directive to **EXTERNAL**. Then specify the commands to be run during backing up and restoration using the **EXTERNAL_BACKUP** and **EXTERNAL_RESTORE** directives. Optionally, also specify the **EXTERNAL_IGNORE_ERRORS** and **EXTERNAL_CHECK** directives. See **/usr/share/rear/conf/default.conf** for an example configuration.

APPENDIX A. CONSISTENT NETWORK DEVICE NAMING

Red Hat Enterprise Linux 6 provides consistent network device naming for network interfaces. This feature changes the name of network interfaces on a system in order to make locating and differentiating the interfaces easier.

Traditionally, network interfaces in Linux are enumerated as **eth[0123...]**, but these names do not necessarily correspond to actual labels on the chassis. Modern server platforms with multiple network adapters can encounter non-deterministic and counter-intuitive naming of these interfaces. This affects both network adapters embedded on the motherboard (*Lan-on-Motherboard*, or *LOM*) and add-in (single and multiport) adapters.

The new naming convention assigns names to network interfaces based on their physical location, whether embedded or in PCI slots. By converting to this naming convention, system administrators will no longer have to guess at the physical location of a network port, or modify each system to rename them into some consistent order.

This feature, implemented via the **biosdevname** program, will change the name of all embedded network interfaces, PCI card network interfaces, and virtual function network interfaces from the existing **eth[0123...]** to the new naming convention as shown in [Table A.1, “The new naming convention”](#).

Table A.1. The new naming convention

Device	Old Name	New Name
Embedded network interface (LOM)	eth[0123...]	em[1234...]^[a]
PCI card network interface	eth[0123...]	p<slot>p<ethernet port>^[b]
Virtual function	eth[0123...]	p<slot>p<ethernet port>_<virtual interface>^[c]

[a] New enumeration starts at **1**.
 [b] For example: **p3p4**
 [c] For example: **p3p4_1**

System administrators may continue to write rules in **/etc/udev/rules.d/70-persistent-net.rules** to change the device names to anything desired; those will take precedence over this physical location naming convention.

A.1. AFFECTED SYSTEMS

Consistent network device naming is enabled by default for a set of **Dell PowerEdge, C Series**, and **Precision Workstation** systems. For more details regarding the impact on Dell systems, visit <https://access.redhat.com/kb/docs/DOC-47318>.

For all other systems, it will be disabled by default; see [Section A.2, “System Requirements”](#) and [Section A.3, “Enabling and Disabling the Feature”](#) for more details.

Regardless of the type of system, Red Hat Enterprise Linux 6 guests running under Red Hat Enterprise Linux 5 hosts will not have devices renamed, since the virtual machine BIOS does not provide SMBIOS information. Upgrades from Red Hat Enterprise Linux 6.0 to Red Hat Enterprise Linux 6.1 are unaffected, and the old **eth[0123...]** naming convention will continue to be used.

A.2. SYSTEM REQUIREMENTS

The **biosdevname** program uses information from the system's BIOS, specifically the *type 9* (System Slot) and *type 41* (Onboard Devices Extended Information) fields contained within the SMBIOS. If the system's BIOS does not have SMBIOS version 2.6 or higher and this data, the new naming convention will not be used. Most older hardware does not support this feature because of a lack of BIOSes with the correct SMBIOS version and field information. For BIOS or SMBIOS version information, contact your hardware vendor.

For this feature to take effect, the **biosdevname** package must also be installed. The **biosdevname** package is part of the **base** package group in Red Hat Enterprise Linux 6. All install options, except for **Minimal Install**, include this package. It is not installed on upgrades of Red Hat Enterprise Linux 6.0 to RHEL 6.1.

A.3. ENABLING AND DISABLING THE FEATURE

To disable the consistent network device naming on Dell systems that would normally have it on by default, pass the following option on the boot command line, both during and after installation:

```
biosdevname=0
```

To enable this feature on other system types that meet the minimum requirements (see [Section A.2, "System Requirements"](#)), pass the following option on the boot command line, both during and after installation:

```
biosdevname=1
```

Unless the system meets the minimum requirements, this option will be ignored and the system will boot with the traditional network interface name format.

If the **biosdevname** install option is specified, it must remain as a boot option for the lifetime of the system.

A.4. NOTES FOR ADMINISTRATORS

Many system customization files can include network interface names, and thus will require updates if moving a system from the old convention to the new convention. If you use the new naming convention, you will also need to update network interface names in areas such as custom iptables rules, scripts altering irqbalance, and other similar configuration files. Also, enabling this change for installation will require modification to existing kickstart files that use device names via the **ksdevice** parameter; these kickstart files will need to be updated to use the network device's MAC address or the network device's new name.

Red Hat strongly recommends that you consider this feature to be an install-time choice; enabling or disabling the feature post-install, while technically possible, can be complicated and is not recommended. For those system administrators who want to do so, on a system that meets the minimum requirements, remove the **/etc/udev/rules.d/70-persistent-net.rules** file and the **HWADDR**

lines from all **/etc/sysconfig/network-scripts/ifcfg-*** files. In addition, rename those **ifcfg-*** files to use this new naming convention. The new names will be in effect after reboot. Remember to update any custom scripts, iptables rules, and service configuration files that might include network interface names.

APPENDIX B. RPM

The *RPM Package Manager* (RPM) is an open packaging system, which runs on Red Hat Enterprise Linux as well as other Linux and UNIX systems. Red Hat, Inc. and the Fedora Project encourage other vendors to use RPM for their own products. RPM is distributed under the terms of the *GPL (GNU General Public License)*.

The RPM Package Manager only works with packages built to work with the *RPM format*. RPM is itself provided as a pre-installed rpm package. For the end user, RPM makes system updates easy. Installing, uninstalling and upgrading RPM packages can be accomplished with short commands. RPM maintains a database of installed packages and their files, so you can invoke powerful queries and verifications on your system.

The RPM package format has been improved for Red Hat Enterprise Linux 6. RPM packages are now compressed using the XZ lossless data compression format, which has the benefit of greater compression and less CPU usage during decompression, and support multiple strong hash algorithms, such as SHA-256, for package signing and verification.



WARNING

For most package management tasks, the **Yum** package manager offers equal and often greater capabilities and utility than RPM. **Yum** also performs and tracks complicated system dependency resolution, and will complain and force system integrity checks if you use RPM as well to install and remove packages. For these reasons, it is highly recommended that you use **Yum** instead of RPM whenever possible to perform package management tasks. See [Chapter 8, Yum](#).

If you prefer a graphical interface, you can use the **PackageKit** GUI application, which uses **Yum** as its back end, to manage your system's packages. See [Chapter 9, PackageKit](#) for details.



IMPORTANT

When installing a package, ensure it is compatible with your operating system and processor architecture. This can usually be determined by checking the package name. Many of the following examples show RPM packages compiled for the AMD64/Intel 64 computer architectures; thus, the RPM file name ends in **x86_64.rpm**.

During upgrades, RPM handles configuration files carefully, so that you never lose your customizations—something that you cannot accomplish with regular **.tar.gz** files.

For the developer, RPM allows you to take software source code and package it into source and binary packages for end users. This process is quite simple and is driven from a single file and optional patches that you create. This clear delineation between *pristine* sources and your patches along with build instructions eases the maintenance of the package as new versions of the software are released.



NOTE

Because RPM makes changes to your system, you must be logged in as root to install, remove, or upgrade an RPM package.

B.1. RPM DESIGN GOALS

To understand how to use RPM, it can be helpful to understand the design goals of RPM:

Upgradability

With RPM, you can upgrade individual components of your system without completely reinstalling. When you get a new release of an operating system based on RPM, such as Red Hat Enterprise Linux, you do not need to reinstall a fresh copy of the operating system your machine (as you might need to with operating systems based on other packaging systems). RPM allows intelligent, fully-automated, in-place upgrades of your system. In addition, configuration files in packages are preserved across upgrades, so you do not lose your customizations. There are no special upgrade files needed to upgrade a package because the same RPM file is used to both install and upgrade the package on your system.

Powerful Querying

RPM is designed to provide powerful querying options. You can perform searches on your entire database for packages or even just certain files. You can also easily find out what package a file belongs to and from where the package came. The files an RPM package contains are in a compressed archive, with a custom binary header containing useful information about the package and its contents, allowing you to query individual packages quickly and easily.

System Verification

Another powerful RPM feature is the ability to verify packages. If you are worried that you deleted an important file for some package, you can verify the package. You are then notified of anomalies, if any—at which point you can reinstall the package, if necessary. Any configuration files that you modified are preserved during reinstallation.

Pristine Sources

A crucial design goal was to allow the use of *pristine* software sources, as distributed by the original authors of the software. With RPM, you have the pristine sources along with any patches that were used, plus complete build instructions. This is an important advantage for several reasons. For instance, if a new version of a program is released, you do not necessarily have to start from scratch to get it to compile. You can look at the patch to see what you *might* need to do. All the compiled-in defaults, and all of the changes that were made to get the software to build properly, are easily visible using this technique.

The goal of keeping sources pristine may seem important only for developers, but it results in higher quality software for end users, too.

B.2. USING RPM

RPM has five basic modes of operation (not counting package building): installing, uninstalling, upgrading, querying, and verifying. This section contains an overview of each mode. For complete details and options, try **rpm --help** or **man rpm**. You can also see [Section B.5, “Additional Resources”](#) for more information on RPM.

B.2.1. Finding RPM Packages

Before using any RPM packages, you must know where to find them. An Internet search returns many RPM repositories, but if you are looking for Red Hat RPM packages, they can be found at the following locations:

- The Red Hat Enterprise Linux installation media contain many installable RPMs.
- The initial RPM repositories provided with the YUM package manager. See [Chapter 8, Yum](#) for details on how to use the official Red Hat Enterprise Linux package repositories.
- The Extra Packages for Enterprise Linux (EPEL) is a community effort to provide high-quality add-on packages for Red Hat Enterprise Linux. See <http://fedoraproject.org/wiki/EPEL> for details on EPEL RPM packages.
- Unofficial, third-party repositories not affiliated with Red Hat also provide RPM packages.



IMPORTANT

When considering third-party repositories for use with your Red Hat Enterprise Linux system, pay close attention to the repository's web site with regard to package compatibility before adding the repository as a package source. Alternate package repositories may offer different, incompatible versions of the same software, including packages already included in the Red Hat Enterprise Linux repositories.

- The Red Hat Errata Page, available at <http://www.redhat.com/apps/support/errata/>.

B.2.2. Installing and Upgrading

RPM packages typically have file names like **tree-1.5.3-2.el6.x86_64.rpm**. The file name includes the package name (**tree**), version (**1.5.3**), release (**2**), operating system major version (**el6**) and CPU architecture (**x86_64**).

You can use **rpm**'s **-U** option to:

- upgrade an existing but older package on the system to a newer version, or
- install the package even if an older version is not already installed.

That is, **rpm -U <rpm_file>** is able to perform the function of either *upgrading* or *installing* as is appropriate for the package.

Assuming the **tree-1.5.3-2.el6.x86_64.rpm** package is in the current directory, log in as root and type the following command at a shell prompt to either upgrade or install the tree package as determined by **rpm**:

```
rpm -Uvh tree-1.5.3-2.el6.x86_64.rpm
```



NOTE

The **-v** and **-h** options (which are combined with **-U**) cause **rpm** to print more verbose output and display a progress meter using hash signs.

If the upgrade/installation is successful, the following output is displayed:

Preparing... 1:tree	##### [100%] ##### [100%]
------------------------	------------------------------



WARNING

rpm provides two different options for installing packages: the aforementioned **-U** option (which historically stands for *upgrade*), and the **-i** option, historically standing for *install*. Because the **-U** option subsumes both install and upgrade functions, we recommend to use **rpm -Uvh** with all packages except kernel packages.

You should always use the **-i** option to *install* a new kernel package instead of upgrading it. This is because using the **-U** option to upgrade a kernel package removes the previous (older) kernel package, which could render the system unable to boot if there is a problem with the new kernel. Therefore, use the **rpm -i <kernel_package>** command to install a new kernel *without* replacing any older kernel packages. For more information on installing kernel packages, see [Chapter 30, Manually Upgrading the Kernel](#).

The signature of a package is checked automatically when installing or upgrading a package. The signature confirms that the package was signed by an authorized party. For example, if the verification of the signature fails, an error message such as the following is displayed:

```
error: tree-1.5.3-2.el6.x86_64.rpm: Header V3 RSA/SHA256 signature: BAD, key ID  
d22e77f2
```

If it is a new, header-only, signature, an error message such as the following is displayed:

```
error: tree-1.5.3-2.el6.x86_64.rpm: Header V3 RSA/SHA256 signature: BAD,  
key ID d22e77f2
```

If you do not have the appropriate key installed to verify the signature, the message contains the word **NOKEY**:

```
warning: tree-1.5.3-2.el6.x86_64.rpm: Header V3 RSA/SHA1 signature: NOKEY, key ID 57bbccba
```

See [Section B.3, “Checking a Package's Signature”](#) for more information on checking a package's signature.

B.2.2.1. Package Already Installed

If a package of the same name and version is already installed, the following output is displayed:

Preparing... package tree-1.5.3-2.el6.x86_64 is already installed	##### [100%]
--	--------------

However, if you want to install the package anyway, you can use the **--replacepkgs** option, which tells RPM to ignore the error:

```
rpm -Uvh --replacepkgs tree-1.5.3-2.el6.x86_64.rpm
```

This option is helpful if files installed from the RPM were deleted or if you want the original configuration files from the RPM to be installed.

B.2.2.2. Conflicting Files

If you attempt to install a package that contains a file which has already been installed by another package, the following is displayed:

```
Preparing... #####
file /usr/bin/foobar from install of foo-1.0-1.el6.x86_64 conflicts
with file from package bar-3.1.1.el6.x86_64
```

To make RPM ignore this error, use the **--replacefiles** option:

```
rpm -Uvh --replacefiles foo-1.0-1.el6.x86_64.rpm
```

B.2.2.3. Unresolved Dependency

RPM packages may sometimes depend on other packages, which means that they require other packages to be installed to run properly. If you try to install a package which has an unresolved dependency, output similar to the following is displayed:

```
error: Failed dependencies:
bar.so.3()(64bit) is needed by foo-1.0-1.el6.x86_64
```

If you are installing a package from the Red Hat Enterprise Linux installation media, such as from a CD-ROM or DVD, the dependencies may be available. Find the suggested package(s) on the Red Hat Enterprise Linux installation media or on one of the active Red Hat Enterprise Linux mirrors and add it to the command:

```
rpm -Uvh foo-1.0-1.el6.x86_64.rpm bar-3.1.1.el6.x86_64.rpm
```

If installation of both packages is successful, output similar to the following is displayed:

```
Preparing... ###### [100%]
1:foo      ##### [ 50%]
2:bar      ##### [100%]
```

You can try the **--whatprovides** option to determine which package contains the required file.

```
rpm -q --whatprovides "bar.so.3"
```

If the package that contains **bar.so.3** is in the RPM database, the name of the package is displayed:

```
bar-3.1.1.el6.i586.rpm
```



WARNING

Although we can *force rpm* to install a package that gives us a **Failed dependencies** error (using the `--nodeps` option), this is *not* recommended, and will usually result in the installed package failing to run. Installing or removing packages with **rpm --nodeps** can cause applications to misbehave and/or crash, and can cause serious package management problems or, possibly, system failure. For these reasons, it is best to heed such warnings; the package manager—whether **RPM**, **Yum** or **PackageKit**—shows us these warnings and suggests possible fixes because accounting for dependencies is critical. The **Yum** package manager can perform dependency resolution and fetch dependencies from online repositories, making it safer, easier and smarter than forcing **rpm** to carry out actions without regard to resolving dependencies.

B.2.3. Configuration File Changes

Because RPM performs intelligent upgrading of packages with configuration files, you may see one or the other of the following messages:

saving /etc/foo.conf as /etc/foo.conf.rpmsave

This message means that changes you made to the configuration file may not be *forward-compatible* with the new configuration file in the package, so RPM saved your original file and installed a new one. You should investigate the differences between the two configuration files and resolve them as soon as possible, to ensure that your system continues to function properly.

Alternatively, RPM may save the package's *new* configuration file as, for example, **foo.conf.rpmnew**, and leave the configuration file you modified untouched. You should still resolve any conflicts between your modified configuration file and the new one, usually by merging changes from the old one to the new one with a **diff** program.

If you attempt to upgrade to a package with an *older* version number (that is, if a higher version of the package is already installed), the output is similar to the following:

package foo-2.0-1.el6.x86_64.rpm (which is newer than foo-1.0-1) is already installed

To force RPM to upgrade anyway, use the **--oldpackage** option:

rpm -Uvh --oldpackage foo-1.0-1.el6.x86_64.rpm

B.2.4. Uninstalling

Uninstalling a package is just as simple as installing one. Type the following command at a shell prompt:

rpm -e foo



NOTE

Notice that we used the package *name* **foo**, not the name of the original package *file*, **foo-1.0-1.el6.x86_64**. If you attempt to uninstall a package using the **rpm -e** command and the original full file name, you will receive a package name error.

You can encounter dependency errors when uninstalling a package if another installed package depends on the one you are trying to remove. For example:

```
rpm -e ghostscript
error: Failed dependencies:
libgs.so.8()(64bit) is needed by (installed) libspectre-0.2.2-3.el6.x86_64
libgs.so.8()(64bit) is needed by (installed) foomatic-4.0.3-1.el6.x86_64
libijs-0.35.so()(64bit) is needed by (installed) gutenprint-5.2.4-5.el6.x86_64
ghostscript is needed by (installed) printer-filters-1.1-4.el6.noarch
```

Similar to how we searched for a shared object library (i.e. a **<library_name>.so.<number>** file) in [Section B.2.2.3, “Unresolved Dependency”](#), we can search for a 64-bit shared object library using this exact syntax (and making sure to quote the file name):

```
~]# rpm -q --whatprovides "libgs.so.8()(64bit)"
ghostscript-8.70-1.el6.x86_64
```



WARNING

Although we can *force* **rpm** to remove a package that gives us a **Failed dependencies** error (using the **--nodeps** option), this is *not* recommended, and may cause harm to other installed applications. Installing or removing packages with **rpm --nodeps** can cause applications to misbehave and/or crash, and can cause serious package management problems or, possibly, system failure. For these reasons, it is best to heed such warnings; the package manager—whether **RPM**, **Yum** or **PackageKit**—shows us these warnings and suggests possible fixes because accounting for dependencies is critical. The **Yum** package manager can perform dependency resolution and fetch dependencies from online repositories, making it safer, easier and smarter than forcing **rpm** to carry out actions without regard to resolving dependencies.

B.2.5. Freshening

Freshening is similar to upgrading, except that only existent packages are upgraded. Type the following command at a shell prompt:

```
rpm -Fvh foo-2.0-1.el6.x86_64.rpm
```

RPM's freshen option checks the versions of the packages specified on the command line against the versions of packages that have already been installed on your system. When a newer version of an already-installed package is processed by RPM's freshen option, it is upgraded to the newer version.

However, RPM's `freshen` option does not install a package if no previously-installed package of the same name exists. This differs from RPM's `upgrade` option, as an `upgrade` does install packages whether or not an older version of the package was already installed.

Freshening works for single packages or package groups. If you have just downloaded a large number of different packages, and you only want to upgrade those packages that are already installed on your system, freshening does the job. Thus, you do not have to delete any unwanted packages from the group that you downloaded before using RPM.

In this case, issue the following with the `*.rpm` glob:

```
rpm -Fvh *.rpm
```

RPM then automatically upgrades only those packages that are already installed.

B.2.6. Querying

The RPM database stores information about all RPM packages installed in your system. It is stored in the directory `/var/lib/rpm/`, and is used to query what packages are installed, what versions each package is, and to calculate any changes to any files in the package since installation, among other use cases.

To query this database, use the `-q` option. The `rpm -q package name` command displays the package name, version, and release number of the installed package `<package_name>`. For example, using `rpm -q tree` to query installed package `tree` might generate the following output:

```
tree-1.5.2.2-4.el6.x86_64
```

You can also use the following *Package Selection Options* (which is a subheading in the RPM man page: see `man rpm` for details) to further refine or qualify your query:

- `-a` – queries all currently installed packages.
- `-f <file_name>` – queries the RPM database for which package owns `<file_name>`. Specify the absolute path of the file (for example, `rpm -qf /bin/ls` instead of `rpm -qf ls`).
- `-p <package_file>` – queries the uninstalled package `<package_file>`.

There are a number of ways to specify what information to display about queried packages. The following options are used to select the type of information for which you are searching. These are called the *Package Query Options*.

- `-i` displays package information including name, description, release, size, build date, install date, vendor, and other miscellaneous information.
- `-l` displays the list of files that the package contains.
- `-s` displays the state of all the files in the package.
- `-d` displays a list of files marked as documentation (man pages, info pages, READMEs, etc.) in the package.
- `-c` displays a list of files marked as configuration files. These are the files you edit after installation to adapt and customize the package to your system (for example, `sendmail.cf`, `passwd`, `inittab`, etc.).

For options that display lists of files, add `-v` to the command to display the lists in a familiar `ls -l` format.

B.2.7. Verifying

Verifying a package compares information about files installed from a package with the same information from the original package. Among other things, verifying compares the file size, MD5 sum, permissions, type, owner, and group of each file.

The command **rpm -V** verifies a package. You can use any of the *Verify Options* listed for querying to specify the packages you want to verify. A simple use of verifying is **rpm -V tree**, which verifies that all the files in the tree package are as they were when they were originally installed. For example:

- To verify a package containing a particular file:

```
rpm -Vf /usr/bin/tree
```

In this example, **/usr/bin/tree** is the absolute path to the file used to query a package.

- To verify ALL installed packages throughout the system (which will take some time):

```
rpm -Va
```

- To verify an installed package against an RPM package file:

```
rpm -Vp tree-1.5.3-2.el6.x86_64.rpm
```

This command can be useful if you suspect that your RPM database is corrupt.

If everything verified properly, there is no output. If there are any discrepancies, they are displayed. The format of the output is a string of eight characters (a "c" denotes a configuration file) and then the file name. Each of the eight characters denotes the result of a comparison of one attribute of the file to the value of that attribute recorded in the RPM database. A single period (.) means the test passed. The following characters denote specific discrepancies:

- **5** – MD5 checksum
- **S** – file size
- **L** – symbolic link
- **T** – file modification time
- **D** – device
- **U** – user
- **G** – group
- **M** – mode (includes permissions and file type)
- **?** – unreadable file (file permission errors, for example)

If you see any output, use your best judgment to determine if you should remove the package, reinstall it, or fix the problem in another way.

B.3. CHECKING A PACKAGE'S SIGNATURE

If you want to verify that a package has not been corrupted or tampered with, examine only the md5sum by typing the following command at a shell prompt (where <rpm_file> is the file name of the RPM package):

```
rpm -K --nosignature <rpm_file>
```

The message **<rpm_file>: rsa sha1 (md5) pgp md5 OK** (specifically the **OK** part of it) is displayed. This brief message means that the file was not corrupted during download. To see a more verbose message, replace **-K** with **-Kvv** in the command.

On the other hand, how trustworthy is the developer who created the package? If the package is *signed* with the developer's GnuPG key, you know that the developer really is who they say they are.

An RPM package can be signed using *GNU Privacy Guard* (or GnuPG), to help you make certain your downloaded package is trustworthy.

GnuPG is a tool for secure communication; it is a complete and free replacement for the encryption technology of PGP, an electronic privacy program. With GnuPG, you can authenticate the validity of documents and encrypt/decrypt data to and from other recipients. GnuPG is capable of decrypting and verifying PGP 5.x files as well.

During installation, GnuPG is installed by default. That way you can immediately start using GnuPG to verify any packages that you receive from Red Hat. Before doing so, you must first import Red Hat's public key.

B.3.1. Importing Keys

To verify Red Hat packages, you must import the Red Hat GnuPG key. To do so, execute the following command at a shell prompt:

```
rpm --import /usr/share/rhn/RPM-GPG-KEY
```

To display a list of all keys installed for RPM verification, execute the command:

```
rpm -qa gpg-pubkey*
```

For the Red Hat key, the output includes:

```
gpg-pubkey-db42a60e-37ea5438
```

To display details about a specific key, use **rpm -qi** followed by the output from the previous command:

```
rpm -qi gpg-pubkey-db42a60e-37ea5438
```

B.3.2. Verifying Signature of Packages

To check the GnuPG signature of an RPM file after importing the builder's GnuPG key, use the following command (replace <rpm-file> with the file name of the RPM package):

```
rpm -K <rpm-file>
```

If all goes well, the following message is displayed: **md5 gpg OK**. This means that the signature of the package has been verified, that it is not corrupt, and therefore is safe to install and use.

B.4. PRACTICAL AND COMMON EXAMPLES OF RPM USAGE

RPM is a useful tool for both managing your system and diagnosing and fixing problems. The best way to make sense of all its options is to look at some examples.

- Perhaps you have deleted some files by accident, but you are not sure what you deleted. To verify your entire system and see what might be missing, you could try the following command:

```
rpm -Va
```

If some files are missing or appear to have been corrupted, you should probably either re-install the package or uninstall and then re-install the package.

- At some point, you might see a file that you do not recognize. To find out which package owns it, enter:

```
rpm -qf /usr/bin/ghostscript
```

The output would look like the following:

```
ghostscript-8.70-1.el6.x86_64
```

- We can combine the above two examples in the following scenario. Say you are having problems with **/usr/bin/paste**. You would like to verify the package that owns that program, but you do not know which package owns **paste**. Enter the following command,

```
rpm -Vf /usr/bin/paste
```

and the appropriate package is verified.

- Do you want to find out more information about a particular program? You can try the following command to locate the documentation which came with the package that owns that program:

```
rpm -qdf /usr/bin/free
```

The output would be similar to the following:

```
/usr/share/doc/procps-3.2.8/BUGS  
/usr/share/doc/procps-3.2.8/FAQ  
/usr/share/doc/procps-3.2.8/NEWS  
/usr/share/doc/procps-3.2.8/TODO  
/usr/share/man/man1/free.1.gz  
/usr/share/man/man1/pgrep.1.gz  
/usr/share/man/man1/pkill.1.gz  
/usr/share/man/man1/pmap.1.gz  
/usr/share/man/man1/ps.1.gz  
/usr/share/man/man1/pwdx.1.gz  
/usr/share/man/man1/skill.1.gz  
/usr/share/man/man1/slabtop.1.gz  
/usr/share/man/man1/snice.1.gz  
/usr/share/man/man1/tload.1.gz  
/usr/share/man/man1/top.1.gz  
/usr/share/man/man1/uptime.1.gz  
/usr/share/man/man1/w.1.gz
```

```
/usr/share/man/man1/watch.1.gz
/usr/share/man/man5/sysctl.conf.5.gz
/usr/share/man/man8/sysctl.8.gz
/usr/share/man/man8/vmstat.8.gz
```

- You may find a new RPM, but you do not know what it does. To find information about it, use the following command:

```
rpm -qip crontabs-1.10-32.1.el6.noarch.rpm
```

The output would be similar to the following:

```
Name      : crontabs          Relocations: (not relocatable)
Version   : 1.10              Vendor: Red Hat, Inc.
Release   : 32.1.el6          Build Date: Thu 03 Dec 2009 02:17:44 AM CET
Install Date: (not installed) Build Host: js20-bc1-11.build.redhat.com
Group     : System Environment/Base Source RPM: crontabs-1.10-32.1.el6.src.rpm
Size      : 2486               License: Public Domain and GPLv2
Signature  : RSA/8, Wed 24 Feb 2010 08:46:13 PM CET, Key ID 938a80caf21541eb
Packager   : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
Summary    : Root crontab files used to schedule the execution of programs
Description :
The crontabs package contains root crontab files and directories.
You will need to install cron daemon to run the jobs from the crontabs.
The cron daemon such as crond or fcron checks the crontab files to
see when particular commands are scheduled to be executed. If commands
are scheduled, it executes them.
Crontabs handles a basic system function, so it should be installed on
your system.
```

- Perhaps you now want to see what files the **crontabs** RPM package installs. You would enter the following:

```
rpm -qlp crontabs-1.10-32.1.el6.noarch.rpm
```

The output is similar to the following:

```
/etc/cron.daily
/etc/cron.hourly
/etc/cron.monthly
/etc/cron.weekly
/etc/crontab
/usr/bin/run-parts
/usr/share/man/man4/crontabs.4.gz
```

These are just a few examples. As you use RPM, you may find more uses for it.

B.5. ADDITIONAL RESOURCES

RPM is an extremely complex utility with many options and methods for querying, installing, upgrading, and removing packages. See the following resources to learn more about RPM.

B.5.1. Installed Documentation

- **rpm --help** – This command displays a quick reference of RPM parameters.
- **man rpm** – The RPM man page gives more detail about RPM parameters than the **rpm --help** command.

B.5.2. Useful Websites

- The RPM website – <http://www.rpm.org/>
- The RPM mailing list can be subscribed to, and its archives read from, here – <https://lists.rpm.org/mailman/listinfo/rpm-list>

APPENDIX C. THE X WINDOW SYSTEM

While the heart of Red Hat Enterprise Linux is the kernel, for many users, the face of the operating system is the graphical environment provided by the *X Window System*, also called *X*.

Other windowing environments have existed in the UNIX world, including some that predate the release of the X Window System in June 1984. Nonetheless, X has been the default graphical environment for most UNIX-like operating systems, including Red Hat Enterprise Linux, for many years.

The graphical environment for Red Hat Enterprise Linux is supplied by the *X.Org Foundation*, an open source organization created to manage development and strategy for the X Window System and related technologies. X.Org is a large-scale, rapid-developing project with hundreds of developers around the world. It features a wide degree of support for a variety of hardware devices and architectures, and runs on myriad operating systems and platforms.

The X Window System uses a client-server architecture. Its main purpose is to provide network transparent window system, which runs on a wide range of computing and graphics machines. The *X server* (the **Xorg** binary) listens for connections from *X client* applications via a network or local loopback interface. The server communicates with the hardware, such as the video card, monitor, keyboard, and mouse. X client applications exist in the user space, creating a *graphical user interface (GUI)* for the user and passing user requests to the X server.

C.1. THE X SERVER

Red Hat Enterprise Linux 6 uses X server version, which includes several video drivers, EXA, and platform support enhancements over the previous release, among others. In addition, this release includes several automatic configuration features for the X server, as well as the generic input driver, **evdev**, that supports all input devices that the kernel knows about, including most mice and keyboards.

X11R7.1 was the first release to take specific advantage of making the X Window System modular. This release split X into logically distinct modules, which make it easier for open source developers to contribute code to the system.

In the current release, all libraries, headers, and binaries live under the **/usr/** directory. The **/etc/X11/** directory contains configuration files for X client and server applications. This includes configuration files for the X server itself, the X display managers, and many other base components.

The configuration file for the newer Fontconfig-based font architecture is still **/etc/fonts/fonts.conf**. For more information on configuring and adding fonts, see [Section C.4, "Fonts"](#).

Because the X server performs advanced tasks on a wide array of hardware, it requires detailed information about the hardware it works on. The X server is able to automatically detect most of the hardware that it runs on and configure itself accordingly. Alternatively, hardware can be manually specified in configuration files.

The Red Hat Enterprise Linux system installer, Anaconda, installs and configures X automatically, unless the X packages are not selected for installation. If there are any changes to the monitor, video card or other devices managed by the X server, most of the time, X detects and reconfigures these changes automatically. In rare cases, X must be reconfigured manually.

C.2. DESKTOP ENVIRONMENTS AND WINDOW MANAGERS

Once an X server is running, X client applications can connect to it and create a GUI for the user. A range of GUIs are available with Red Hat Enterprise Linux, from the rudimentary *Tab Window Manager* (twm) to the highly developed and interactive desktop environment (such as *GNOME* or *KDE*) that most

Red Hat Enterprise Linux users are familiar with.

To create the latter, more comprehensive GUI, two main classes of X client application must connect to the X server: a *window manager* and a *desktop environment*.

C.2.1. Maximum number of concurrent GUI sessions

Multiple GUI sessions for different users can be run at the same time on the same machine. The maximum number of concurrent GUI sessions is limited by the hardware, especially by the memory size, and by the workload demands of the running applications. For common PCs the maximum possible number of concurrent GUI sessions is not higher than 10 to 15, depending on previously described circumstances. Logging the same user into GNOME more than once on the same machine is not supported, because some applications could terminate unexpectedly.

C.2.2. Desktop Environments

A desktop environment integrates various X clients to create a common graphical user environment and a development platform.

Desktop environments have advanced features allowing X clients and other running processes to communicate with one another, while also allowing all applications written to work in that environment to perform advanced tasks, such as drag-and-drop operations.

Red Hat Enterprise Linux provides two desktop environments:

- *GNOME* – The default desktop environment for Red Hat Enterprise Linux based on the GTK+ 2 graphical toolkit.
- *KDE* – An alternative desktop environment based on the Qt 4 graphical toolkit.

Both GNOME and KDE have advanced-productivity applications, such as word processors, spreadsheets, and Web browsers; both also provide tools to customize the look and feel of the GUI. Additionally, if both the GTK+ 2 and the Qt libraries are present, KDE applications can run in GNOME and vice versa.

C.2.3. Window Managers

Window managers are X client programs which are either part of a desktop environment or, in some cases, stand-alone. Their primary purpose is to control the way graphical windows are positioned, resized, or moved. Window managers also control title bars, window focus behavior, and user-specified key and mouse button bindings.

The Red Hat Enterprise Linux repositories provide five different window managers.

metacity

The *Metacity* window manager is the default window manager for GNOME. It is a simple and efficient window manager which supports custom themes. This window manager is automatically pulled in as a dependency when the GNOME desktop is installed.

kwin

The *KWin* window manager is the default window manager for KDE. It is an efficient window manager which supports custom themes. This window manager is automatically pulled in as a dependency when the KDE desktop is installed.

compiz

The *Compiz* compositing window manager is based on OpenGL and can use 3D graphics hardware to create fast compositing desktop effects for window management. Advanced features, such as a cube workspace, are implemented as loadable plug-ins. To run this window manager, you need to install the compiz package.

mwm

The *Motif Window Manager* (**mwm**) is a basic, stand-alone window manager. Since it is designed to be stand-alone, it should not be used in conjunction with GNOME or KDE. To run this window manager, you need to install the openmotif package.

twm

The minimalist *Tab Window Manager* (**twm**), which provides the most basic tool set among the available window managers, can be used either as a stand-alone or with a desktop environment. To run this window manager, you need to install the xorg-x11-twm package.

C.3. X SERVER CONFIGURATION FILES

The X server is a single binary executable **/usr/bin/Xorg**; a symbolic link **X** pointing to this file is also provided. Associated configuration files are stored in the **/etc/X11/** and **/usr/share/X11/** directories.

The X Window System supports two different configuration schemes. Configuration files in the **xorg.conf.d** directory contain preconfigured settings from vendors and from distribution, and these files should not be edited by hand. Configuration in the **xorg.conf** file, on the other hand, is done completely by hand but is not necessary in most scenarios.



NOTE

All necessary parameters for a display and peripherals are auto-detected and configured during installation. The configuration file for the X server, **/etc/X11/xorg.conf**, that was necessary in previous releases, is not supplied with the current release of the X Window System. It can still be useful to create the file manually to configure new hardware, to set up an environment with multiple video cards, or for debugging purposes.

The **/usr/lib/xorg/modules/** (or **/usr/lib64/xorg/modules/**) directory contains X server modules that can be loaded dynamically at runtime. By default, only some modules in **/usr/lib/xorg/modules/** are automatically loaded by the X server.

When Red Hat Enterprise Linux 6 is installed, the configuration files for X are created using information gathered about the system hardware during the installation process by the HAL (Hardware Abstraction Layer) configuration back end. Whenever the X server is started, it asks HAL for the list of input devices and adds each of them with their respective driver. Whenever a new input device is plugged in, or an existing input device is removed, HAL notifies the X server about the change. Because of this notification system, devices using the **mouse**, **kbd**, or **vmmouse** driver configured in the **xorg.conf** file are, by default, ignored by the X server. See [Section C.3.3.3, “The ServerFlags section”](#) for further details. Additional configuration is provided in the **/etc/X11/xorg.conf.d/** directory and it can override or augment any configuration that has been obtained through HAL.

C.3.1. The Structure of the Configuration

The format of the X configuration files is comprised of many different sections which address specific aspects of the system hardware. Each section begins with a **Section "section-name"** line, where

"*section-name*" is the title for the section, and ends with an **EndSection** line. Each section contains lines that include option names and one or more option values. Some of these are sometimes enclosed in double quotes ("").

Some options within the **/etc/X11/xorg.conf** file accept a Boolean switch which turns the feature on or off. The acceptable values are:

- **1, on, true, or yes** – Turns the option on.
- **0, off, false, or no** – Turns the option off.

The following shows a typical configuration file for the keyboard. Lines beginning with a hash sign (#) are not read by the X server and are used for human-readable comments.

```
# This file is autogenerated by system-setup-keyboard. Any
# modifications will be lost.

Section "InputClass"
    Identifier "system-setup-keyboard"
    MatchIsKeyboard "on"
    Option "XkbModel" "pc105"
    Option "XkbLayout" "cz,us"
    # Option "XkbVariant" "(null)"
    Option "XkbOptions" "terminate:ctrl_alt_bksp,grp:shifts_toggle,grp_led:scroll"
EndSection
```

C.3.2. The **xorg.conf.d** Directory

The X server supports two configuration directories. The **/usr/share/X11/xorg.conf.d** provides separate configuration files from vendors or third-party packages; changes to files in this directory may be overwritten by settings specified in the **/etc/X11/xorg.conf** file. The **/etc/X11/xorg.conf.d** directory stores user-specific configuration.

Files with the suffix **.conf** in configuration directories are parsed by the X server upon startup and are treated like part of the traditional **xorg.conf** configuration file. These files may contain one or more sections; for a description of the options in a section and the general layout of the configuration file, see [Section C.3.3, “The **xorg.conf** File”](#) or to the **xorg.conf(5)** man page. The X server essentially treats the collection of configuration files as one big file with entries from **xorg.conf** at the end. Users are encouraged to put custom configuration into **/etc/xorg.conf** and leave the directory for configuration snippets provided by the distribution.

C.3.3. The **xorg.conf** File

In previous releases of the X Window System, **/etc/X11/xorg.conf** file was used to store initial setup for X. When a change occurred with the monitor, video card or other device managed by the X server, the file needed to be edited manually. In Red Hat Enterprise Linux, there is rarely a need to manually create and edit the **/etc/X11/xorg.conf** file. Nevertheless, it is still useful to understand various sections and optional parameters available, especially when troubleshooting or setting up unusual hardware configuration.

In the following, some important sections are described in the order in which they appear in a typical **/etc/X11/xorg.conf** file. More detailed information about the X server configuration file can be found in the **xorg.conf(5)** man page. This section is mostly intended for advanced users as most configuration options described below are not needed in typical configuration scenarios.

C.3.3.1. The InputClass section

InputClass is a new type of configuration section that does not apply to a single device but rather to a class of devices, including hot-plugged devices. An **InputClass** section's scope is limited by the matches specified; in order to apply to an input device, all matches must apply to the device as seen in the example below:

```
Section "InputClass"
    Identifier      "touchpad catchall"
    MatchIsTouchpad "on"
    Driver         "synaptics"
EndSection
```

If this snippet is present in an **xorg.conf** file or an **xorg.conf.d** directory, any touchpad present in the system is assigned the **synaptics** driver.



NOTE

Note that due to alphanumeric sorting of configuration files in the **xorg.conf.d** directory, the **Driver** setting in the example above overwrites previously set driver options. The more generic the class, the earlier it should be listed.

The match options specify which devices a section may apply to. To match a device, all match options must correspond. The following options are commonly used in the **InputClass** section:

- **MatchIsPointer**, **MatchIsKeyboard**, **MatchIsTouchpad**, **MatchIsTouchscreen**, **MatchIsJoystick** – Boolean options to specify a type of device.
- **MatchProduct "*product_name*"** – this option matches if the *product_name* substring occurs in the product name of the device.
- **MatchVendor "*vendor_name*"** – this option matches if the *vendor_name* substring occurs in the vendor name of the device.
- **MatchDevicePath "*/path/to/device*"** – this option matches any device if its device path corresponds to the patterns given in the "*/path/to/device*" template, for example **/dev/input/event***. See the **fnmatch(3)** man page for further details.
- **MatchTag "*tag_pattern*"** – this option matches if at least one tag assigned by the HAL configuration back end matches the *tag_pattern* pattern.

A configuration file may have multiple **InputClass** sections. These sections are optional and are used to configure a class of input devices as they are automatically added. An input device can match more than one **InputClass** section. When arranging these sections, it is recommended to put generic matches above specific ones because each input class can override settings from a previous one if an overlap occurs.

C.3.3.2. The InputDevice section

Each **InputDevice** section configures one input device for the X server. Previously, systems typically had at least one **InputDevice** section for the keyboard, and most mouse settings were automatically detected.

With Red Hat Enterprise Linux 6, no **InputDevice** configuration is needed for most setups, and the `xorg-x11-drv-*` input driver packages provide the automatic configuration through HAL. The default driver for both keyboards and mice is **evdev**.

The following example shows a typical **InputDevice** section for a keyboard:

```
Section "InputDevice"
Identifier "Keyboard0"
Driver "kbd"
Option "XkbModel" "pc105"
Option "XkbLayout" "us"
EndSection
```

The following entries are commonly used in the **InputDevice** section:

- **Identifier** – Specifies a unique name for this **InputDevice** section. This is a required entry.
- **Driver** – Specifies the name of the device driver X must load for the device. If the **AutoAddDevices** option is enabled (which is the default setting), any input device section with **Driver "mouse"** or **Driver "kbd"** will be ignored. This is necessary due to conflicts between the legacy mouse and keyboard drivers and the new **evdev** generic driver. Instead, the server will use the information from the back end for any input devices. Any custom input device configuration in the **xorg.conf** should be moved to the back end. In most cases, the back end will be HAL and the configuration location will be the **/etc/X11/xorg.conf.d** directory.
- **Option** – Specifies necessary options pertaining to the device.

A mouse may also be specified to override any auto-detected values for the device. The following options are typically included when adding a mouse in the **xorg.conf** file:

- **Protocol** – Specifies the protocol used by the mouse, such as **IMPS/2**.
- **Device** – Specifies the location of the physical device.
- **Emulate3Buttons** – Specifies whether to allow a two-button mouse to act like a three-button mouse when both mouse buttons are pressed simultaneously.

Consult the **xorg.conf(5)** man page for a complete list of valid options for this section.

C.3.3.3. The **ServerFlags** section

The optional **ServerFlags** section contains miscellaneous global X server settings. Any settings in this section may be overridden by options placed in the **ServerLayout** section (see [Section C.3.3.4, “The ServerLayout Section”](#) for details).

Each entry within the **ServerFlags** section occupies a single line and begins with the term **Option** followed by an option enclosed in double quotation marks ("").

The following is a sample **ServerFlags** section:

```
Section "ServerFlags"
Option "DontZap" "true"
EndSection
```

The following lists some of the most useful options:

- **"DontZap" "boolean"** – When the value of <boolean> is set to **true**, this setting prevents the use of the **Ctrl+Alt+Backspace** key combination to immediately terminate the X server.



NOTE

Even if this option is enabled, the key combination still must be configured in the X Keyboard Extension (XKB) map before it can be used. One way how to add the key combination to the map is to run the following command:

```
setxkbmap -option "terminate:ctrl_alt_bksp"
```

- **"DontZoom" "boolean"** – When the value of <boolean> is set to **true**, this setting prevents cycling through configured video resolutions using the **Ctrl+Alt+Keypad-Plus** and **Ctrl+Alt+Keypad-Minus** key combinations.
- **"AutoAddDevices" "boolean"** – When the value of <boolean> is set to **false**, the server will not hot plug input devices and instead rely solely on devices configured in the **xorg.conf** file. See [Section C.3.3.2, "The InputDevice section"](#) for more information concerning input devices. This option is enabled by default and HAL (hardware abstraction layer) is used as a back end for device discovery.

C.3.3.4. The ServerLayout Section

The **ServerLayout** section binds together the input and output devices controlled by the X server. At a minimum, this section must specify one input device and one output device. By default, a monitor (output device) and a keyboard (input device) are specified.

The following example shows a typical **ServerLayout** section:

```
Section "ServerLayout"
    Identifier "Default Layout"
    Screen 0 "Screen0" 0 0
    InputDevice "Mouse0" "CorePointer"
    InputDevice "Keyboard0" "CoreKeyboard"
EndSection
```

The following entries are commonly used in the **ServerLayout** section:

- **Identifier** – Specifies a unique name for this **ServerLayout** section.
- **Screen** – Specifies the name of a **Screen** section to be used with the X server. More than one **Screen** option may be present.

The following is an example of a typical **Screen** entry:

```
Screen 0 "Screen0" 0 0
```

The first number in this example **Screen** entry (**0**) indicates that the first monitor connector, or *head* on the video card, uses the configuration specified in the **Screen** section with the identifier **"Screen0"**.

An example of a **Screen** section with the identifier **"Screen0"** can be found in [Section C.3.3.8, "The Screen section"](#).

If the video card has more than one head, another **Screen** entry with a different number and a different **Screen** section identifier is necessary.

The numbers to the right of "**Screen0**" give the absolute X and Y coordinates for the upper left corner of the screen (**0 0** by default).

- **InputDevice** – Specifies the name of an **InputDevice** section to be used with the X server.

It is advisable that there be at least two **InputDevice** entries: one for the default mouse and one for the default keyboard. The options **CorePointer** and **CoreKeyboard** indicate that these are the primary mouse and keyboard. If the **AutoAddDevices** option is enabled, this entry needs not to be specified in the **ServerLayout** section. If the **AutoAddDevices** option is disabled, both mouse and keyboard are auto-detected with the default values.

- **Option "option-name"** – An optional entry which specifies extra parameters for the section. Any options listed here override those listed in the **ServerFlags** section.

Replace <option-name> with a valid option listed for this section in the **xorg.conf(5)** man page.

It is possible to put more than one **ServerLayout** section in the **/etc/X11/xorg.conf** file. By default, the server only reads the first one it encounters, however. If there is an alternative **ServerLayout** section, it can be specified as a command-line argument when starting an X session; as in the **Xorg -layout <layoutname>** command.

C.3.3.5. The **Files** section

The **Files** section sets paths for services vital to the X server, such as the font path. This is an optional section, as these paths are normally detected automatically. This section can be used to override automatically detected values.

The following example shows a typical **Files** section:

```
Section "Files"
    RgbPath "/usr/share/X11/rgb.txt"
    FontPath "unix/:7100"
EndSection
```

The following entries are commonly used in the **Files** section:

- **ModulePath** – An optional parameter which specifies alternate directories which store X server modules.

C.3.3.6. The **Monitor** section

Each **Monitor** section configures one type of monitor used by the system. This is an optional entry as most monitors are now detected automatically.

This example shows a typical **Monitor** section for a monitor:

```
Section "Monitor"
    Identifier "Monitor0"
    VendorName "Monitor Vendor"
    ModelName "DDC Probed Monitor - ViewSonic G773-2"
    DisplaySize 320 240
```

```

HorizSync 30.0 - 70.0
VertRefresh 50.0 - 180.0
EndSection

```

The following entries are commonly used in the **Monitor** section:

- **Identifier** – Specifies a unique name for this **Monitor** section. This is a required entry.
- **VendorName** – An optional parameter which specifies the vendor of the monitor.
- **ModelName** – An optional parameter which specifies the monitor's model name.
- **DisplaySize** – An optional parameter which specifies, in millimeters, the physical size of the monitor's picture area.
- **HorizSync** – Specifies the range of horizontal sync frequencies compatible with the monitor, in kHz. These values help the X server determine the validity of built-in or specified **Modeline** entries for the monitor.
- **VertRefresh** – Specifies the range of vertical refresh frequencies supported by the monitor, in kHz. These values help the X server determine the validity of built-in or specified **Modeline** entries for the monitor.
- **Modeline** – An optional parameter which specifies additional video modes for the monitor at particular resolutions, with certain horizontal sync and vertical refresh resolutions. See the **xorg.conf(5)** man page for a more detailed explanation of **Modeline** entries.
- **Option "option-name"** – An optional entry which specifies extra parameters for the section. Replace <option-name> with a valid option listed for this section in the **xorg.conf(5)** man page.

C.3.3.7. The Device section

Each **Device** section configures one video card on the system. While one **Device** section is the minimum, additional instances may occur for each video card installed on the machine.

The following example shows a typical **Device** section for a video card:

```

Section "Device"
Identifier "Videocard0"
Driver "mga"
VendorName "Videocard vendor"
BoardName "Matrox Millennium G200"
VideoRam 8192
Option "dpms"
EndSection

```

The following entries are commonly used in the **Device** section:

- **Identifier** – Specifies a unique name for this **Device** section. This is a required entry.
- **Driver** – Specifies which driver the X server must load to utilize the video card. A list of drivers can be found in **/usr/share/hwdata/videodrivers**, which is installed with the hwdata package.
- **VendorName** – An optional parameter which specifies the vendor of the video card.
- **BoardName** – An optional parameter which specifies the name of the video card.

- **VideoRam** – An optional parameter which specifies the amount of RAM available on the video card, in kilobytes. This setting is only necessary for video cards the X server cannot probe to detect the amount of video RAM.
- **BusID** – An entry which specifies the bus location of the video card. On systems with only one video card a **BusID** entry is optional and may not even be present in the default **/etc/X11/xorg.conf** file. On systems with more than one video card, however, a **BusID** entry is required.
- **Screen** – An optional entry which specifies which monitor connector or head on the video card the **Device** section configures. This option is only useful for video cards with multiple heads.

If multiple monitors are connected to different heads on the same video card, separate **Device** sections must exist and each of these sections must have a different **Screen** value.

Values for the **Screen** entry must be an integer. The first head on the video card has a value of **0**. The value for each additional head increments this value by one.

- **Option "option-name"** – An optional entry which specifies extra parameters for the section. Replace <option-name> with a valid option listed for this section in the **xorg.conf(5)** man page.

One of the more common options is "**dpms**" (for Display Power Management Signaling, a VESA standard), which activates the Energy Star energy compliance setting for the monitor.

C.3.3.8. The Screen section

Each **Screen** section binds one video card (or video card head) to one monitor by referencing the **Device** section and the **Monitor** section for each. While one **Screen** section is the minimum, additional instances may occur for each video card and monitor combination present on the machine.

The following example shows a typical **Screen** section:

```
Section "Screen"
    Identifier "Screen0"
    Device "Videocard0"
    Monitor "Monitor0"
    DefaultDepth 16

    SubSection "Display"
        Depth 24
        Modes "1280x1024" "1280x960" "1152x864" "1024x768" "800x600" "640x480"
    EndSubSection

    SubSection "Display"
        Depth 16
        Modes "1152x864" "1024x768" "800x600" "640x480"
    EndSubSection
EndSection
```

The following entries are commonly used in the **Screen** section:

- **Identifier** – Specifies a unique name for this **Screen** section. This is a required entry.
- **Device** – Specifies the unique name of a **Device** section. This is a required entry.

- **Monitor** – Specifies the unique name of a **Monitor** section. This is only required if a specific **Monitor** section is defined in the **xorg.conf** file. Normally, monitors are detected automatically.
- **DefaultDepth** – Specifies the default color depth in bits. In the previous example, **16** (which provides thousands of colors) is the default. Only one **DefaultDepth** entry is permitted, although this can be overridden with the Xorg command-line option **-depth <n>**, where **<n>** is any additional depth specified.
- **SubSection "Display"** – Specifies the screen modes available at a particular color depth. The **Screen** section can have multiple **Display** subsections, which are entirely optional since screen modes are detected automatically.

This subsection is normally used to override auto-detected modes.

- **Option "option-name"** – An optional entry which specifies extra parameters for the section. Replace **<option-name>** with a valid option listed for this section in the **xorg.conf(5)** man page.

C.3.3.9. The DRI section

The optional **DRI** section specifies parameters for the *Direct Rendering Infrastructure (DRI)*. DRI is an interface which allows 3D software applications to take advantage of 3D hardware acceleration capabilities built into most modern video hardware. In addition, DRI can improve 2D performance via hardware acceleration, if supported by the video card driver.

This section is rarely used, as the DRI Group and Mode are automatically initialized to default values. If a different Group or Mode is needed, then adding this section to the **xorg.conf** file will override the default values.

The following example shows a typical **DRI** section:

```
Section "DRI"
    Group 0
    Mode 0666
EndSection
```

Since different video cards use DRI in different ways, do not add to this section without first referring to <http://dri.freedesktop.org/wiki/>.

C.4. FONTS

Red Hat Enterprise Linux uses *Fontconfig* subsystem to manage and display fonts under the X Window System. It simplifies font management and provides advanced display features, such as anti-aliasing. This system is used automatically for applications programmed using the **Qt 3** or **GTK+ 2** graphical toolkits, or their newer versions.

The Fontconfig font subsystem allows applications to directly access fonts on the system and use the *X FreeType interface library (Xft)* or other rendering mechanisms to render Fontconfig fonts with advanced features such as anti-aliasing. Graphical applications can use the Xft library with Fontconfig to draw text to the screen.



NOTE

Fontconfig uses the **/etc/fonts/fonts.conf** configuration file, which should not be edited by hand.

**WARNING**

Any system where the user expects to run remote X applications needs to have the **fonts** group installed. This can be done by selecting the group in the installer, and also by running the **yum groupinstall fonts** command after installation.

C.4.1. Adding Fonts to Fontconfig

Adding new fonts to the Fontconfig subsystem is a straightforward process:

1. To add fonts for an individual user, copy the new fonts into the **.fonts/** directory in the user's home directory.

To add fonts system-wide, copy the new fonts into the **/usr/share/fonts/** directory. It is a good idea to create a new subdirectory, such as **local/** or similar, to help distinguish between user-installed and default fonts.

2. Run the **fc-cache** command as root to update the font information cache:

```
fc-cache <path-to-font-directory>
```

In this command, replace **<path-to-font-directory>** with the directory containing the new fonts (either **/usr/share/fonts/local/** or **/home/<user>/fonts/**).

**NOTE**

Individual users may also install fonts interactively, by typing **fonts:///** into the **Nautilus** address bar, and dragging the new font files there.

C.5. RUNLEVELS AND X

In most cases, the Red Hat Enterprise Linux installer configures a machine to boot into a graphical login environment, known as *runlevel 5*. It is possible, however, to boot into a text-only multi-user mode called *runlevel 3* and begin an X session from there.

The following subsections review how X starts up in both runlevel 3 and runlevel 5. For more information about runlevels, see [Section 12.1, “Configuring the Default Runlevel”](#).

C.5.1. Runlevel 3

When in runlevel 3, the best way to start an X session is to log in and type **startx**. The **startx** command is a front-end to the **xinit** command, which launches the X server (**Xorg**) and connects X client applications to it. Because the user is already logged into the system at runlevel 3, **startx** does not launch a display manager or authenticate users. See [Section C.5.2, “Runlevel 5”](#) for more information about display managers.

1. When the **startx** command is executed, it searches for the **.xinitrc** file in the user's home directory to define the desktop environment and possibly other X client applications to run. If no **.xinitrc** file is present, it uses the system default **/etc/X11/xinit/xinitrc** file instead.

2. The default **xinitrc** script then searches for user-defined files and default system files, including **.Xresources**, **.Xmodmap**, and **.Xkbmap** in the user's home directory, and **Xresources**, **Xmodmap**, and **Xkbmap** in the **/etc/X11/** directory. The **Xmodmap** and **Xkbmap** files, if they exist, are used by the **xmodmap** utility to configure the keyboard. The **Xresources** file is read to assign specific preference values to applications.
3. After setting the above options, the **xinitrc** script executes all scripts located in the **/etc/X11/xinit/xinitrc.d/** directory. One important script in this directory is **xinput.sh**, which configures settings such as the default language.
4. The **xinitrc** script attempts to execute **.Xclients** in the user's home directory and turns to **/etc/X11/xinit/Xclients** if it cannot be found. The purpose of the **Xclients** file is to start the desktop environment or, possibly, just a basic window manager. The **.Xclients** script in the user's home directory starts the user-specified desktop environment in the **.Xclients-default** file. If **.Xclients** does not exist in the user's home directory, the standard **/etc/X11/xinit/Xclients** script attempts to start another desktop environment, trying GNOME first, then KDE, followed by **twm**.

When in runlevel 3, the user is returned to a text mode user session after ending an X session.

C.5.2. Runlevel 5

When the system boots into runlevel 5, a special X client application called a *display manager* is launched. A user must authenticate using the display manager before any desktop environment or window managers are launched.

Depending on the desktop environments installed on the system, three different display managers are available to handle user authentication.

- **GDM** (GNOME Display Manager) – The default display manager for Red Hat Enterprise Linux. **GNOME** allows the user to configure language settings, shutdown, restart or log in to the system.
- **KDM** – KDE's display manager which allows the user to shutdown, restart or log in to the system.
- **xdm** (X Window Display Manager) – A very basic display manager which only lets the user log in to the system.

When booting into runlevel 5, the **/etc/X11/prefdm** script determines the preferred display manager by referencing the **/etc/sysconfig/desktop** file. A list of options for this file is available in this file:

```
/usr/share/doc/initscripts-<version-number>/sysconfig.txt
```

where **<version-number>** is the version number of the initscripts package.

Each of the display managers reference the **/etc/X11/xdm/Xsetup_0** file to set up the login screen. Once the user logs into the system, the **/etc/X11/xdm/GiveConsole** script runs to assign ownership of the console to the user. Then, the **/etc/X11/xdm/Xsession** script runs to accomplish many of the tasks normally performed by the **xinitrc** script when starting X from runlevel 3, including setting system and user resources, as well as running the scripts in the **/etc/X11/xinit/xinitrc.d/** directory.

Users can specify which desktop environment they want to use when they authenticate using the **GNOME** or **KDE** display managers by selecting it from the **Sessions** menu item accessed by selecting **System → Preferences → More Preferences → Sessions**. If the desktop environment is not specified in the display manager, the **/etc/X11/xdm/Xsession** script checks the **.xsession** and **.Xclients** files in the

user's home directory to decide which desktop environment to load. As a last resort, the **/etc/X11/xinit/Xclients** file is used to select a desktop environment or window manager to use in the same way as runlevel 3.

When the user finishes an X session on the default display (**:0**) and logs out, the **/etc/X11/xdm/TakeConsole** script runs and reassigned ownership of the console to the root user. The original display manager, which continues running after the user logged in, takes control by spawning a new display manager. This restarts the X server, displays a new login window, and starts the entire process over again.

The user is returned to the display manager after logging out of X from runlevel 5.

For more information on how display managers control user authentication, see the **/usr/share/doc/gdm-<version-number>/README**, where <version-number> is the version number for the gdm package installed, or the **xdm** man page.

C.6. ACCESSING GRAPHICAL APPLICATIONS REMOTELY

It is possible to access graphical applications on a remote server using these methods:

- You can start a separate application directly from your SSH session in your local X server. For that, you need to enable X11 forwarding. See [Section 14.5.1, “X11 Forwarding”](#) for details.
- You can run the whole X session over network using VNC. This method can be useful, especially when you are using a workstation without X server, for example, a non-Linux system. See [Chapter 15, TigerVNC](#) for details.

C.7. ADDITIONAL RESOURCES

There is a large amount of detailed information available about the X server, the clients that connect to it, and the assorted desktop environments and window managers.

C.7.1. Installed Documentation

- **/usr/share/X11/doc/** – contains detailed documentation on the X Window System architecture, as well as how to get additional information about the Xorg project as a new user.
- **/usr/share/doc/gdm-<version-number>/README** – contains information on how display managers control user authentication.
- **man xorg.conf** – Contains information about the **xorg.conf** configuration files, including the meaning and syntax for the different sections within the files.
- **man Xorg** – Describes the **Xorg** display server.

C.7.2. Useful Websites

- <http://www.X.org/> – Home page of the X.Org Foundation, which produces major releases of the X Window System bundled with Red Hat Enterprise Linux to control the necessary hardware and provide a GUI environment.
- <http://dri.sourceforge.net/> – Home page of the DRI (Direct Rendering Infrastructure) project. The DRI is the core hardware 3D acceleration component of X.
- <http://www.gnome.org/> – Home of the GNOME project.

- <http://www.kde.org/> – Home of the KDE desktop environment.

APPENDIX D. THE SYSCONFIG DIRECTORY

This appendix outlines some of the files and directories found in the `/etc/sysconfig/` directory, their function, and their contents. The information in this appendix is not intended to be complete, as many of these files have a variety of options that are only used in very specific or rare circumstances.



NOTE

The actual content of your `/etc/sysconfig/` directory depends on the programs you have installed on your machine. To find the name of the package the configuration file belongs to, type the following at a shell prompt:

```
~]$ yum provides /etc/sysconfig/filename
```

See [Section 8.2.4, “Installing Packages”](#) for more information on how to install new packages in Red Hat Enterprise Linux.

D.1. FILES IN THE /ETC/SYSCONFIG/ DIRECTORY

The following sections offer descriptions of files normally found in the `/etc/sysconfig/` directory.

D.1.1. /etc/sysconfig/arpwatch

The `/etc/sysconfig/arpwatch` file is used to pass arguments to the `arpwatch` daemon at boot time. By default, it contains the following option:

OPTIONS=value

Additional options to be passed to the `arpwatch` daemon. For example:

```
OPTIONS="-u arpwatch -e root -s 'root (Arpwatch)'"
```

D.1.2. /etc/sysconfig/authconfig

The `/etc/sysconfig/authconfig` file sets the authorization to be used on the host. By default, it contains the following options:

USEMKHOMEDIR=boolean

A Boolean to enable (**yes**) or disable (**no**) creating a home directory for a user on the first login. For example:

```
USEMKHOMEDIR=no
```

USEPAMACCESS=boolean

A Boolean to enable (**yes**) or disable (**no**) the PAM authentication. For example:

```
USEPAMACCESS=no
```

USESSSDAUTH=boolean

A Boolean to enable (**yes**) or disable (**no**) the SSSD authentication. For example:

```
USESSSSDAUTH=no
```

USESHADOW=*boolean*

A Boolean to enable (**yes**) or disable (**no**) shadow passwords. For example:

```
USESHADOW=yes
```

USEWINBIND=*boolean*

A Boolean to enable (**yes**) or disable (**no**) using Winbind for user account configuration. For example:

```
USEWINBIND=no
```

USEDDB=*boolean*

A Boolean to enable (**yes**) or disable (**no**) the FAS authentication. For example:

```
USEDDB=no
```

USEFPRINTD=*boolean*

A Boolean to enable (**yes**) or disable (**no**) the fingerprint authentication. For example:

```
USEFPRINTD=yes
```

FORCESMARTCARD=*boolean*

A Boolean to enable (**yes**) or disable (**no**) enforcing the smart card authentication. For example:

```
FORCESMARTCARD=no
```

PASSWDALGORITHM=*value*

The password algorithm. The *value* can be **bigcrypt**, **descrypt**, **md5**, **sha256**, or **sha512**. For example:

```
PASSWDALGORITHM=sha512
```

USELDAPAUTH=*boolean*

A Boolean to enable (**yes**) or disable (**no**) the LDAP authentication. For example:

```
USELDAPAUTH=no
```

USELOCAUTHORIZE=*boolean*

A Boolean to enable (**yes**) or disable (**no**) the local authorization for local users. For example:

```
USELOCAUTHORIZE=yes
```

USECRACKLIB=*boolean*

A Boolean to enable (**yes**) or disable (**no**) using the CrackLib. For example:

USECRACKLIB=yes

USEWINBINDAUTH=boolean

A Boolean to enable (**yes**) or disable (**no**) the Winbind authentication. For example:

USEWINBINDAUTH=no

USESSMARTCARD=boolean

A Boolean to enable (**yes**) or disable (**no**) the smart card authentication. For example:

USESSMARTCARD=no

USELDAP=boolean

A Boolean to enable (**yes**) or disable (**no**) using LDAP for user account configuration. For example:

USELDAP=no

USENIS=boolean

A Boolean to enable (**yes**) or disable (**no**) using NIS for user account configuration. For example:

USENIS=no

USEKERBEROS=boolean

A Boolean to enable (**yes**) or disable (**no**) the Kerberos authentication. For example:

USEKERBEROS=no

USESYSNETAUTH=boolean

A Boolean to enable (**yes**) or disable (**no**) authenticating system accounts with network services. For example:

USESYSNETAUTH=no

USESMBAUTH=boolean

A Boolean to enable (**yes**) or disable (**no**) the SMB authentication. For example:

USESMBAUTH=no

USESSSD=boolean

A Boolean to enable (**yes**) or disable (**no**) using SSSD for obtaining user information. For example:

USESSSD=no

USEHESIOD=boolean

A Boolean to enable (**yes**) or disable (**no**) using the Hesiod name service. For example:

```
USEHESIOD=no
```

See [Chapter 13, Configuring Authentication](#) for more information on this topic.

D.1.3. /etc/sysconfig/autofs

The **/etc/sysconfig/autofs** file defines custom options for the automatic mounting of devices. This file controls the operation of the automount daemons, which automatically mount file systems when you use them and unmount them after a period of inactivity. File systems can include network file systems, CD-ROM drives, diskettes, and other media.

By default, it contains the following options:

MASTER_MAP_NAME=value

The default name for the master map. For example:

```
MASTER_MAP_NAME="auto.master"
```

TIMEOUT=value

The default mount timeout. For example:

```
TIMEOUT=300
```

NEGATIVE_TIMEOUT=value

The default negative timeout for unsuccessful mount attempts. For example:

```
NEGATIVE_TIMEOUT=60
```

MOUNT_WAIT=value

The time to wait for a response from **mount**. For example:

```
MOUNT_WAIT=-1
```

UMOUNT_WAIT=value

The time to wait for a response from **umount**. For example:

```
UMOUNT_WAIT=12
```

BROWSE_MODE=boolean

A Boolean to enable (**yes**) or disable (**no**) browsing the maps. For example:

```
BROWSE_MODE="no"
```

MOUNT_NFS_DEFAULT_PROTOCOL=value

The default protocol to be used by **mount.nfs**. For example:

```
MOUNT_NFS_DEFAULT_PROTOCOL=4
```

APPEND_OPTIONS=*boolean*

A Boolean to enable (**yes**) or disable (**no**) appending the global options instead of replacing them.
For example:

```
APPEND_OPTIONS="yes"
```

LOGGING=*value*

The default logging level. The *value* has to be either **none**, **verbose**, or **debug**. For example:

```
LOGGING="none"
```

LDAP_URI=*value*

A space-separated list of server URIs in the form of **protocol://server**. For example:

```
LDAP_URI="ldaps://ldap.example.com/"
```

LDAP_TIMEOUT=*value*

The synchronous API calls timeout. For example:

```
LDAP_TIMEOUT=-1
```

LDAP_NETWORK_TIMEOUT=*value*

The network response timeout. For example:

```
LDAP_NETWORK_TIMEOUT=8
```

SEARCH_BASE=*value*

The base Distinguished Name (DN) for the map search. For example:

```
SEARCH_BASE=""
```

AUTH_CONF_FILE=*value*

The default location of the SASL authentication configuration file. For example:

```
AUTH_CONF_FILE="/etc/autofs_ldap_auth.conf"
```

MAP_HASH_TABLE_SIZE=*value*

The hash table size for the map cache. For example:

```
MAP_HASH_TABLE_SIZE=1024
```

USE_MISC_DEVICE=*boolean*

A Boolean to enable (**yes**) or disable (**no**) using the autofs miscellaneous device. For example:

```
USE_MISC_DEVICE="yes"
```

OPTIONS=value

Additional options to be passed to the LDAP daemon. For example:

```
OPTIONS=""
```

D.1.4. /etc/sysconfig/clock

The **/etc/sysconfig/clock** file controls the interpretation of values read from the system hardware clock. It is used by the **Date/Time Properties** tool, and should not be edited by hand. By default, it contains the following option:

ZONE=value

The time zone file under **/usr/share/zoneinfo** that **/etc/localtime** is a copy of. For example:

```
ZONE="Europe/Prague"
```

See [Section 2.1, “Date/Time Properties Tool”](#) for more information on the **Date/Time Properties** tool and its usage.

D.1.5. /etc/sysconfig/dhcpd

The **/etc/sysconfig/dhcpd** file is used to pass arguments to the **dhcpd** daemon at boot time. By default, it contains the following options:

DHCPDARGS=value

Additional options to be passed to the **dhcpd** daemon. For example:

```
DHCPDARGS=
```

See [Chapter 16, DHCP Servers](#) for more information on DHCP and its usage.

D.1.6. /etc/sysconfig/firstboot

The **/etc/sysconfig/firstboot** file defines whether to run the **firstboot** utility. By default, it contains the following option:

RUN_FIRSTBOOT=boolean

A Boolean to enable (**YES**) or disable (**NO**) running the **firstboot** program. For example:

```
RUN_FIRSTBOOT=NO
```

The first time the system boots, the **init** program calls the **/etc/rc.d/init.d/firstboot** script, which looks for the **/etc/sysconfig/firstboot** file. If this file does not contain the **RUN_FIRSTBOOT=NO** option, the **firstboot** program is run, guiding a user through the initial configuration of the system.

**NOTE**

To start the **firstboot** program the next time the system boots, change the value of **RUN_FIRSTBOOT** option to **YES**, and type the following at a shell prompt:

```
~]# chkconfig firstboot on
```

D.1.7. /etc/sysconfig/i18n

The **/etc/sysconfig/i18n** configuration file defines the default language, any supported languages, and the default system font. By default, it contains the following options:

LANG=value

The default language. For example:

```
LANG="en_US.UTF-8"
```

SUPPORTED=value

A colon-separated list of supported languages. For example:

```
SUPPORTED="en_US.UTF-8:en_US:en"
```

SYSFONT=value

The default system font. For example:

```
SYSFONT="latarcyrheb-sun16"
```

D.1.8. /etc/sysconfig/init

The **/etc/sysconfig/init** file controls how the system appears and functions during the boot process. By default, it contains the following options:

BOOTUP=value

The bootup style. The value has to be either **color** (the standard color boot display), **verbose** (an old style display which provides more information), or anything else for the new style display, but without ANSI formatting. For example:

```
BOOTUP=color
```

RES_COL=value

The number of the column in which the status labels start. For example:

```
RES_COL=60
```

MOVE_TO_COL=value

The terminal sequence to move the cursor to the column specified in **RES_COL** (see above). For example:

MOVE_TO_COL="echo -en \\033[\${RES_COL}G"

SETCOLOR_SUCCESS=value

The terminal sequence to set the success color. For example:

SETCOLOR_SUCCESS="echo -en \\033[0;32m"

SETCOLOR_FAILURE=value

The terminal sequence to set the failure color. For example:

SETCOLOR_FAILURE="echo -en \\033[0;31m"

SETCOLOR_WARNING=value

The terminal sequence to set the warning color. For example:

SETCOLOR_WARNING="echo -en \\033[0;33m"

SETCOLOR_NORMAL=value

The terminal sequence to set the default color. For example:

SETCOLOR_NORMAL="echo -en \\033[0;39m"

LOGLEVEL=value

The initial console logging level. The *value* has to be in the range from **1** (kernel panics only) to **8** (everything, including the debugging information). For example:

LOGLEVEL=3

PROMPT=boolean

A Boolean to enable (**yes**) or disable (**no**) the hotkey interactive startup. For example:

PROMPT=yes

AUTOSWAP=boolean

A Boolean to enable (**yes**) or disable (**no**) probing for devices with swap signatures. For example:

AUTOSWAP=no

ACTIVE_CONSOLES=value

The list of active consoles. For example:

ACTIVE_CONSOLES=/dev/tty[1-6]

SINGLE=value

The single-user mode type. The *value* has to be either **/sbin/sulogin** (a user will be prompted for a password to log in), or **/sbin/sushell** (the user will be logged in directly). For example:

SINGLE=/sbin/sushell

D.1.9. /etc/sysconfig/ip6tables-config

The **/etc/sysconfig/ip6tables-config** file stores information used by the kernel to set up IPv6 packet filtering at boot time or whenever the **ip6tables** service is started. Note that you should not modify it unless you are familiar with **ip6tables** rules. By default, it contains the following options:

IP6TABLES_MODULES=value

A space-separated list of helpers to be loaded after the firewall rules are applied. For example:

IP6TABLES_MODULES="ip_nat_ftp ip_nat irc"

IP6TABLES_MODULES_UNLOAD=boolean

A Boolean to enable (**yes**) or disable (**no**) module unloading when the firewall is stopped or restarted. For example:

IP6TABLES_MODULES_UNLOAD="yes"

IP6TABLES_SAVE_ON_STOP=boolean

A Boolean to enable (**yes**) or disable (**no**) saving the current firewall rules when the firewall is stopped. For example:

IP6TABLES_SAVE_ON_STOP="no"

IP6TABLES_SAVE_ON_RESTART=boolean

A Boolean to enable (**yes**) or disable (**no**) saving the current firewall rules when the firewall is restarted. For example:

IP6TABLES_SAVE_ON_RESTART="no"

IP6TABLES_SAVE_COUNTER=boolean

A Boolean to enable (**yes**) or disable (**no**) saving the rule and chain counters. For example:

IP6TABLES_SAVE_COUNTER="no"

IP6TABLES_STATUS_NUMERIC=boolean

A Boolean to enable (**yes**) or disable (**no**) printing IP addresses and port numbers in a numeric format in the status output. For example:

IP6TABLES_STATUS_NUMERIC="yes"

IP6TABLES_STATUS_VERBOSE=boolean

A Boolean to enable (**yes**) or disable (**no**) printing information about the number of packets and bytes in the status output. For example:

IP6TABLES_STATUS_VERBOSE="no"

IP6TABLES_STATUS_LINENUMBERS=boolean

A Boolean to enable (**yes**) or disable (**no**) printing line numbers in the status output. For example:

```
| IP6TABLES_STATUS_LINENUMBERS="yes"
```

**NOTE**

You can create the rules manually using the **ip6tables** command. Once created, type the following at a shell prompt:

```
| ~]# service ip6tables save
```

This will add the rules to **/etc/sysconfig/ip6tables**. Once this file exists, any firewall rules saved in it persist through a system reboot or a service restart.

D.1.10. /etc/sysconfig/kernel

The **/etc/sysconfig/kernel** configuration file controls the kernel selection at boot by using these two options:

UPDATEDDEFAULT=yes

This option makes a newly installed kernel as the default in the boot entry selection.

DEFAULTKERNEL=kernel

This option specifies what package type will be used as the default.

D.1.10.1. Keeping an old kernel version as the default

To keep an old kernel version as the default in the boot entry selection:

- Comment out the UPDATEDDEFAULT option in **/etc/sysconfig/kernel** as follows:

```
| # UPDATEDDEFAULT=yes
```

D.1.10.2. Setting a kernel debugger as the default kernel

To set kernel debugger as the default kernel in boot entry selection:

- Edit the **/etc/sysconfig/kernel** configuration file as follows:

```
| DEFAULTKERNEL=kernel-debug
```

D.1.11. /etc/sysconfig/keyboard

The **/etc/sysconfig/keyboard** file controls the behavior of the keyboard. By default, it contains the following options:

KEYTABLE=value

The name of a keytable file. The files that can be used as keytables start in the **/lib/kbd/keymaps/i386/** directory, and branch into different keyboard layouts from there, all labeled **value.kmap.gz**. The first file name that matches the **KEYTABLE** setting is used. For example:

```
KEYTABLE="us"
```

MODEL=value

The keyboard model. For example:

```
MODEL="pc105+inet"
```

LAYOUT=value

The keyboard layout. For example:

```
LAYOUT="us"
```

KEYBOARDDTYPE=value

The keyboard type. Allowed values are **pc** (a PS/2 keyboard), or **sun** (a Sun keyboard). For example:

```
KEYBOARDDTYPE="pc"
```

D.1.12. /etc/sysconfig/ldap

The **/etc/sysconfig/ldap** file holds the basic configuration for the LDAP server. By default, it contains the following options:

SLAPD_OPTIONS=value

Additional options to be passed to the **slapd** daemon. For example:

```
SLAPD_OPTIONS="-4"
```

SLURPD_OPTIONS=value

Additional options to be passed to the **slurpd** daemon. For example:

```
SLURPD_OPTIONS=""
```

SLAPD_LDAP=boolean

A Boolean to enable (**yes**) or disable (**no**) using the LDAP over TCP (that is, **ldap://**). For example:

```
SLAPD_LDAP="yes"
```

SLAPD_LDAP=boolean

A Boolean to enable (**yes**) or disable (**no**) using the LDAP over IPC (that is, **ldapi://**). For example:

```
SLAPD_LDAP="no"
```

SLAPD_LDAPS=boolean

A Boolean to enable (**yes**) or disable (**no**) using the LDAP over TLS (that is, **ldaps:///**). For example:

```
SLAPD_LDAPS="no"
```

SLAPD_URLS=value

A space-separated list of URLs. For example:

```
SLAPD_URLS="ldapi:///var/lib/ldap_root/ldapi ldapi:/// ldaps:///"
```

SLAPD_SHUTDOWN_TIMEOUT=value

The time to wait for **slapd** to shut down. For example:

```
SLAPD_SHUTDOWN_TIMEOUT=3
```

SLAPD_ULIMIT_SETTINGS=value

The parameters to be passed to **ulimit** before the **slapd** daemon is started. For example:

```
SLAPD_ULIMIT_SETTINGS=""
```

See [Section 20.1, “OpenLDAP”](#) for more information on LDAP and its configuration.

D.1.13. /etc/sysconfig/named

The **/etc/sysconfig/named** file is used to pass arguments to the **named** daemon at boot time. By default, it contains the following options:

ROOTDIR=value

The chroot environment under which the **named** daemon runs. The *value* has to be a full directory path. For example:

```
ROOTDIR="/var/named/chroot"
```

Note that the chroot environment has to be configured first (type **info chroot** at a shell prompt for more information).

OPTIONS=value

Additional options to be passed to **named**. For example:

```
OPTIONS="-6"
```

Note that you should not use the **-t** option. Instead, use **ROOTDIR** as described above.

KEYTAB_FILE=value

The keytab file name. For example:

```
KEYTAB_FILE="/etc/named.keytab"
```

See [Section 17.2, “BIND”](#) for more information on the BIND DNS server and its configuration.

D.1.14. /etc/sysconfig/network

The **/etc/sysconfig/network** file is used to specify information about the desired network configuration. By default, it contains the following options:

NETWORKING=boolean

A Boolean to enable (**yes**) or disable (**no**) networking. For example:

```
NETWORKING=yes
```

HOSTNAME=value

The host name of the machine. For example:

```
HOSTNAME=penguin.example.com
```

The file may also contain some of the following options:

GATEWAY=value

The IP address of the network's gateway. For example:

```
GATEWAY=192.168.1.1
```

This is used as the default gateway when there is no **GATEWAY** directive in an interface's **ifcfg** file.

NM_BOND_VLAN_ENABLED=boolean

A Boolean to allow (**yes**) or disallow (**no**) the **NetworkManager** application from detecting and managing bonding, bridging, and VLAN interfaces. For example:

```
NM_BOND_VLAN_ENABLED=yes
```

The **NM_CONTROLLED** directive is dependent on this option.



NOTE

If you want to completely disable IPv6, you should add these lines to **/etc/sysctl.conf**:

```
net.ipv6.conf.all.disable_ipv6=1
```

```
net.ipv6.conf.default.disable_ipv6=1
```

In addition, adding **ipv6.disable=1** to the kernel command line will disable the kernel module **net-pf-10** which implements IPv6.



WARNING

Do not use custom init scripts to configure network settings. When performing a post-boot network service restart, custom init scripts configuring network settings that are run outside of the network init script lead to unpredictable results.

D.1.15. /etc/sysconfig/ntp

The **/etc/sysconfig/ntp** file is used to pass arguments to the **ntpd** daemon at boot time. By default, it contains the following option:

OPTIONS=value

Additional options to be passed to **ntpd**. For example:

```
OPTIONS="-u ntp:ntp -p /var/run/ntp.pid -g"
```

See [Section 2.1.2, “Network Time Protocol Properties”](#) or [Section 2.2.2, “Network Time Protocol Setup”](#) for more information on how to configure the **ntpd** daemon.

D.1.16. /etc/sysconfig/quagga

The **/etc/sysconfig/quagga** file holds the basic configuration for Quagga daemons. By default, it contains the following options:

QCONFDIR=value

The directory with the configuration files for Quagga daemons. For example:

```
QCONFDIR="/etc/quagga"
```

BGPD_OPTS=value

Additional options to be passed to the **bgpd** daemon. For example:

```
BGPD_OPTS="-A 127.0.0.1 -f ${QCONFDIR}/bgpd.conf"
```

OSPF6D_OPTS=value

Additional options to be passed to the **ospf6d** daemon. For example:

```
OSPF6D_OPTS="-A ::1 -f ${QCONFDIR}/ospf6d.conf"
```

OSPFD_OPTS=value

Additional options to be passed to the **ospf6d** daemon. For example:

```
OSPFD_OPTS="-A 127.0.0.1 -f ${QCONFDIR}/ospf6d.conf"
```

RIPD_OPTS=value

Additional options to be passed to the **ripd** daemon. For example:

```
RIPD_OPTS="-A 127.0.0.1 -f ${QCONFDIR}/ripd.conf"
```

RIPNGD_OPTS=value

Additional options to be passed to the **ripngd** daemon. For example:

```
RIPNGD_OPTS="-A ::1 -f ${QCONFDIR}/ripngd.conf"
```

ZEBRA_OPTS=value

Additional options to be passed to the **zebra** daemon. For example:

```
ZEBRA_OPTS="-A 127.0.0.1 -f ${QCONFDIR}/zebra.conf"
```

ISISD_OPTS=value

Additional options to be passed to the **isisd** daemon. For example:

```
ISISD_OPTS="-A ::1 -f ${QCONFDIR}/isisd.conf"
```

WATCH_OPTS=value

Additional options to be passed to the **watchquagga** daemon. For example:

```
WATCH_OPTS="-Az -b_ -r/sbin/service_%s_restart -s/sbin/service_%s_start -k/sbin/service_%s_stop"
```

WATCH_DAEMONS=value

A space separated list of monitored daemons. For example:

```
WATCH_DAEMONS="zebra bgpd ospfd ospf6d ripd ripngd"
```

D.1.17. /etc/sysconfig/radvd

The **/etc/sysconfig/radvd** file is used to pass arguments to the **radvd** daemon at boot time. By default, it contains the following option:

OPTIONS=value

Additional options to be passed to the **radvd** daemon. For example:

```
OPTIONS="-u radvd"
```

D.1.18. /etc/sysconfig/samba

The **/etc/sysconfig/samba** file is used to pass arguments to the Samba daemons at boot time. By default, it contains the following options:

SMBDOPTIONS=value

Additional options to be passed to **smbd**. For example:

SMBDOPTIONS="-D"

NMBDOPTIONS=*value*

Additional options to be passed to **nmbd**. For example:

NMBDOPTIONS="-D"

WINBINDOPTIONS=*value*

Additional options to be passed to **winbindd**. For example:

WINBINDOPTIONS=""

See [Section 21.1, “Samba”](#) for more information on Samba and its configuration.

D.1.19. /etc/sysconfig/saslauthd

The **/etc/sysconfig/saslauthd** file is used to control which arguments are passed to **saslauthd**, the SASL authentication server. By default, it contains the following options:

SOCKETDIR=*value*

The directory for the **saslauthd**'s listening socket. For example:

SOCKETDIR=/var/run/saslauthd

MECH=*value*

The authentication mechanism to use to verify user passwords. For example:

MECH=pam

DAEMONOPTS=*value*

Options to be passed to the **daemon()** function that is used by the **/etc/rc.d/init.d/saslauthd** init script to start the **saslauthd** service. For example:

DAEMONOPTS="--user saslauth"

FLAGS=*value*

Additional options to be passed to the **saslauthd** service. For example:

FLAGS=

D.1.20. /etc/sysconfig/selinux

The **/etc/sysconfig/selinux** file contains the basic configuration options for SELinux. It is a symbolic link to **/etc/selinux/config**, and by default, it contains the following options:

SELINUX=*value*

The security policy. The *value* can be either **enforcing** (the security policy is always enforced), **permissive** (instead of enforcing the policy, appropriate warnings are displayed), or **disabled** (no policy is used). For example:

```
SELINUX=enforcing
```

SELINUXTYPE=*value*

The protection type. The *value* can be either **targeted** (the targeted processes are protected), or **mls** (the Multi Level Security protection). For example:

```
SELINUXTYPE=targeted
```

D.1.21. /etc/sysconfig/sendmail

The **/etc/sysconfig/sendmail** is used to set the default values for the **Sendmail** application. By default, it contains the following values:

DAEMON=*boolean*

A Boolean to enable (**yes**) or disable (**no**) running **sendmail** as a daemon. For example:

```
DAEMON=yes
```

QUEUE=*value*

The interval at which the messages are to be processed. For example:

```
QUEUE=1h
```

See [Section 19.3.2, “Sendmail”](#) for more information on Sendmail and its configuration.

D.1.22. /etc/sysconfig/spamassassin

The **/etc/sysconfig/spamassassin** file is used to pass arguments to the **spamd** daemon (a daemonized version of **Spamassassin**) at boot time. By default, it contains the following option:

SPAMDOPTIONS=*value*

Additional options to be passed to the **spamd** daemon. For example:

```
SPAMDOPTIONS="-d -c -m5 -H"
```

See [Section 19.4.2.6, “Spam Filters”](#) for more information on Spamassassin and its configuration.

D.1.23. /etc/sysconfig/squid

The **/etc/sysconfig/squid** file is used to pass arguments to the **squid** daemon at boot time. By default, it contains the following options:

SQUID_OPTS=*value*

Additional options to be passed to the **squid** daemon. For example:

```
SQUID_OPTS=""
```

SQUID_SHUTDOWN_TIMEOUT=value

The time to wait for **squid** daemon to shut down. For example:

```
SQUID_SHUTDOWN_TIMEOUT=100
```

SQUID_CONF=value

The default configuration file. For example:

```
SQUID_CONF="/etc/squid/squid.conf"
```

D.1.24. /etc/sysconfig/system-config-users

The **/etc/sysconfig/system-config-users** file is the configuration file for the **User Manager** utility, and should not be edited by hand. By default, it contains the following options:

FILTER=boolean

A Boolean to enable (**true**) or disable (**false**) filtering of system users. For example:

```
FILTER=true
```

ASSIGN_HIGHEST_UID=boolean

A Boolean to enable (**true**) or disable (**false**) assigning the highest available UID to newly added users. For example:

```
ASSIGN_HIGHEST_UID=true
```

ASSIGN_HIGHEST_GID=boolean

A Boolean to enable (**true**) or disable (**false**) assigning the highest available GID to newly added groups. For example:

```
ASSIGN_HIGHEST_GID=true
```

PREFER_SAME_UID_GID=boolean

A Boolean to enable (**true**) or disable (**false**) using the same UID and GID for newly added users when possible. For example:

```
PREFER_SAME_UID_GID=true
```

See [Section 3.2, “Managing Users via the User Manager Application”](#) for more information on **User Manager** and its usage.

D.1.25. /etc/sysconfig/vncservers

The **/etc/sysconfig/vncservers** file configures the way the *Virtual Network Computing* (VNC) server starts up. By default, it contains the following options:

VNCSERVERS=value

A list of space separated **display:username** pairs. For example:

```
VNCSERVERS="2:myusername"
```

VNCERVERARGS[display]=value

Additional arguments to be passed to the VNC server running on the specified *display*. For example:

```
VNCERVERARGS[2]="-geometry 800x600 -nolisten tcp -localhost"
```

D.1.26. /etc/sysconfig/xinetd

The **/etc/sysconfig/xinetd** file is used to pass arguments to the **xinetd** daemon at boot time. By default, it contains the following options:

EXTRAOPTIONS=value

Additional options to be passed to **xinetd**. For example:

```
EXTRAOPTIONS=""
```

XINETD_LANG=value

The locale information to be passed to every service started by **xinetd**. Note that to remove locale information from the **xinetd** environment, you can use an empty string ("") or **none**. For example:

```
XINETD_LANG="en_US"
```

See [Chapter 12, Services and Daemons](#) for more information on how to configure the **xinetd** services.

D.2. DIRECTORIES IN THE /ETC/SYSCONFIG/ DIRECTORY

The following directories are normally found in **/etc/sysconfig/**.

/etc/sysconfig/cbq/

This directory contains the configuration files needed to do *Class Based Queuing* for bandwidth management on network interfaces. CBQ divides user traffic into a hierarchy of classes based on any combination of IP addresses, protocols, and application types.

/etc/sysconfig/networking/

This directory is used by the now deprecated **Network Administration Tool (system-config-network)**, and its contents should not be edited manually. For more information about configuring network interfaces using graphical configuration tools, see [Chapter 10, NetworkManager](#).

/etc/sysconfig/network-scripts/

This directory contains the following network-related configuration files:

- Network configuration files for each configured network interface, such as **ifcfg-eth0** for the **eth0** Ethernet interface.
- Scripts used to bring network interfaces up and down, such as **ifup** and **ifdown**.
- Scripts used to bring ISDN interfaces up and down, such as **ifup-isdn** and **ifdown-isdn**.
- Various shared network function scripts which should not be edited directly.

For more information on the **/etc/sysconfig/network-scripts/** directory, see [Chapter 11, Network Interfaces](#).

/etc/sysconfig/rhn/

This directory contains the configuration files and GPG keys for Red Hat Network. No files in this directory should be edited by hand. For more information on Red Hat Network, see the Red Hat Network website online at <https://rhn.redhat.com/>.

D.3. ADDITIONAL RESOURCES

This chapter is only intended as an introduction to the files in the **/etc/sysconfig/** directory. The following source contains more comprehensive information.

D.3.1. Installed Documentation

/usr/share/doc/initscripts-version/sysconfig.txt

A more authoritative listing of the files found in the **/etc/sysconfig/** directory and the configuration options available for them.

APPENDIX E. THE PROC FILE SYSTEM

The Linux kernel has two primary functions: to control access to physical devices on the computer and to schedule when and how processes interact with these devices. The **/proc/** directory (also called the **proc** file system) contains a hierarchy of special files which represent the current state of the kernel, allowing applications and users to peer into the kernel's view of the system.

The **/proc/** directory contains a wealth of information detailing system hardware and any running processes. In addition, some of the files within **/proc/** can be manipulated by users and applications to communicate configuration changes to the kernel.



NOTE

Later versions of the 2.6 kernel have made the **/proc/ide/** and **/proc/pci/** directories obsolete. The **/proc/ide/** file system is now superseded by files in **sysfs**; to retrieve information on PCI devices, use **lspci** instead. For more information on **sysfs** or **lspci**, see their respective **man** pages.

E.1. A VIRTUAL FILE SYSTEM

Linux systems store all data as *files*. Most users are familiar with the two primary types of files: text and binary. But the **/proc/** directory contains another type of file called a *virtual file*. As such, **/proc/** is often referred to as a *virtual file system*.

Virtual files have unique qualities. Most of them are listed as zero bytes in size, but can still contain a large amount of information when viewed. In addition, most of the time and date stamps on virtual files reflect the current time and date, indicative of the fact they are constantly updated.

Virtual files such as **/proc/interrupts**, **/proc/meminfo**, **/proc/mounts**, and **/proc/partitions** provide an up-to-the-moment glimpse of the system's hardware. Others, like the **/proc/filesystems** file and the **/proc/sys/** directory provide system configuration information and interfaces.

For organizational purposes, files containing information on a similar topic are grouped into virtual directories and sub-directories. Process directories contain information about each running process on the system.

E.1.1. Viewing Virtual Files

Most files within **/proc/** files operate similarly to text files, storing useful system and hardware data in human-readable text format. As such, you can use **cat**, **more**, or **less** to view them. For example, to display information about the system's CPU, run **cat /proc/cpuinfo**. This will return output similar to the following:

```
processor : 0
vendor_id : AuthenticAMD
cpu family : 5
model : 9
model name : AMD-K6(tm) 3D+
Processor stepping : 1 cpu
MHz : 400.919
cache size : 256 KB
fdiv_bug : no
hlt_bug : no
f00f_bug : no
```

```

coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 1
wp : yes
flags : fpu vme de pse tsc msr mce cx8 pge mmx syscall 3dnow k6_mtrr
bogomips : 799.53

```

Some files in **/proc** contain information that is not human-readable. To retrieve information from such files, use tools such as **lspci**, **apm**, **free**, and **top**.



NOTE

Some of the virtual files in the **/proc** directory are readable only by the root user.

E.1.2. Changing Virtual Files

As a general rule, most virtual files within the **/proc** directory are read-only. However, some can be used to adjust settings in the kernel. This is especially true for files in the **/proc/sys** subdirectory.

To change the value of a virtual file, use the following command:

```
echo value > /proc/file
```

For example, to change the host name on the fly, run:

```
echo www.example.com > /proc/sys/kernel/hostname
```

Other files act as binary or Boolean switches. Typing **cat /proc/sys/net/ipv4/ip_forward** returns either a **0** (off or false) or a **1** (on or true). A **0** indicates that the kernel is not forwarding network packets. To turn packet forwarding on, run **echo 1 > /proc/sys/net/ipv4/ip_forward**.



NOTE

Another command used to alter settings in the **/proc/sys** subdirectory is **/sbin/sysctl**. For more information on this command, see [Section E.4, "Using the sysctl Command"](#)

For a listing of some of the kernel configuration files available in the **/proc/sys** subdirectory, see [Section E.3.9, "/proc/sys/"](#).

E.2. TOP-LEVEL FILES WITHIN THE PROC FILE SYSTEM

Below is a list of some of the more useful virtual files in the top-level of the **/proc** directory.



NOTE

In most cases, the content of the files listed in this section are not the same as those installed on your machine. This is because much of the information is specific to the hardware on which Red Hat Enterprise Linux is running for this documentation effort.

E.2.1. /proc/buddyinfo

The **/proc/buddyinfo** file is used primarily for diagnosing memory fragmentation issues. The output depends on the memory layout used, which is architecture specific. The following is an example from a 32-bit system:

```
Node 0, zone DMA 90 6 2 1 1 ...
Node 0, zone Normal 1650 310 5 0 0 ...
Node 0, zone HighMem 2 0 0 1 1 ...
```

Using the buddy algorithm, each column represents the number of memory pages of a certain order, a certain size, that are available at any given time. In the example above, for zone DMA, there are 90 of $2^0 * \text{PAGE_SIZE}$ bytes large chunks of memory. Similarly, there are 6 of $2^1 * \text{PAGE_SIZE}$ chunks and 2 of $2^2 * \text{PAGE_SIZE}$ chunks of memory available.

The **DMA** row references the first 16 MB of memory on the system, the **HighMem** row references all memory greater than 896 MB on the system, and the **Normal** row references the memory in between.

On a 64-bit system, the output might look as follows:

```
Node 0, zone DMA 0 3 1 2 4 3 1 2 3 3 1
Node 0, zone DMA32 295 25850 7065 1645 835 220 78 6 0 1 0
Node 0, zone Normal 3824 3359 736 159 31 3 1 1 1 1 0
```

The **DMA** row references the first 16 MB of memory on the system, the **DMA32** row references all memory allocated for devices that cannot address memory greater than 4 GB, and the **Normal** row references all memory above the **DMA32** allocation, which includes all memory above 4 GB on the system.

E.2.2. /proc/cmdline

This file shows the parameters passed to the kernel at the time it is started. A sample **/proc/cmdline** file looks like the following:

```
ro root=/dev/VolGroup00/LogVol00 rhgb quiet 3
```

This tells us that the kernel is mounted read-only (signified by **(ro)**), located on the first logical volume (**LogVol00**) of the first volume group (**/dev/VolGroup00**). **LogVol00** is the equivalent of a disk partition in a non-LVM system (Logical Volume Management), just as **/dev/VolGroup00** is similar in concept to **/dev/hda1**, but much more extensible.

For more information on LVM used in Red Hat Enterprise Linux, see <http://www.tldp.org/HOWTO/LVM-HOWTO/index.html>.

Next, **rhgb** signals that the **rhgb** package has been installed, and graphical booting is supported, assuming **/etc/inittab** shows a default runlevel set to **id:5:initdefault:**.

Finally, **quiet** indicates all verbose kernel messages are suppressed at boot time.

E.2.3. /proc/cpuinfo

This virtual file identifies the type of processor used by your system. The following is an example of the output typical of **/proc/cpuinfo**:

```
processor : 0
vendor_id : GenuineIntel
```

```

cpu family : 15
model : 2
model name : Intel(R) Xeon(TM) CPU 2.40GHz
stepping : 7 cpu
MHz : 2392.371
cache size : 512 KB
physical id : 0
siblings : 2
runqueue : 0
fdiv_bug : no
hlt_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 2
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi
mmx fxsr sse sse2 ss ht tm
bogomips : 4771.02

```

- **processor** – Provides each processor with an identifying number. On systems that have one processor, only a **0** is present.
- **cpu family** – Authoritatively identifies the type of processor in the system. For an Intel-based system, place the number in front of "86" to determine the value. This is particularly helpful for those attempting to identify the architecture of an older system such as a 586, 486, or 386. Because some RPM packages are compiled for each of these particular architectures, this value also helps users determine which packages to install.
- **model name** – Displays the common name of the processor, including its project name.
- **cpu MHz** – Shows the precise speed in megahertz for the processor to the thousandths decimal place.
- **cache size** – Displays the amount of level 2 memory cache available to the processor.
- **siblings** – Displays the total number of sibling CPUs on the same physical CPU for architectures which use hyper-threading.
- **flags** – Defines a number of different qualities about the processor, such as the presence of a floating point unit (FPU) and the ability to process MMX instructions.

E.2.4. /proc/crypto

This file lists all installed cryptographic ciphers used by the Linux kernel, including additional details for each. A sample **/proc/crypto** file looks like the following:

```

name      : sha1
module    : kernel
type      : digest
blocksize : 64
digestsize : 20
name      : md5
module    : md5

```

```
type      : digest
blocksize : 64
digestsize : 16
```

E.2.5. /proc/devices

This file displays the various character and block devices currently configured (not including devices whose modules are not loaded). Below is a sample output from this file:

Character devices:

```
1 mem
4 /dev/vc/0
4 tty
4 ttyS
5 /dev/tty
5 /dev/console
5 /dev/ptmx
7 vcs
10 misc
13 input
29 fb
36 netlink
128 ptm
136 pts
180 usb
```

Block devices:

```
1 ramdisk
3 ide0
9 md
22 ide1
253 device-mapper
254 mdp
```

The output from **/proc/devices** includes the major number and name of the device, and is broken into two major sections: **Character devices** and **Block devices**.

Character devices are similar to *block devices*, except for two basic differences:

1. Character devices do not require buffering. Block devices have a buffer available, allowing them to order requests before addressing them. This is important for devices designed to store information – such as hard drives – because the ability to order the information before writing it to the device allows it to be placed in a more efficient order.
2. Character devices send data with no preconfigured size. Block devices can send and receive information in blocks of a size configured per device.

For more information about devices, see the **devices.txt** file in the `kernel-doc` package (see [Section E.5, "Additional Resources"](#)).

E.2.6. /proc/dma

This file contains a list of the registered ISA DMA channels in use. A sample **/proc/dma** files looks like the following:

4: cascade

E.2.7. /proc/execdomains

This file lists the *execution domains* currently supported by the Linux kernel, along with the range of personalities they support.

0-0 Linux [kernel]

Think of execution domains as the "personality" for an operating system. Because other binary formats, such as Solaris, UnixWare, and FreeBSD, can be used with Linux, programmers can change the way the operating system treats system calls from these binaries by changing the personality of the task. Except for the **PER_LINUX** execution domain, different personalities can be implemented as dynamically loadable modules.

E.2.8. /proc/fb

This file contains a list of frame buffer devices, with the frame buffer device number and the driver that controls it. Typical output of **/proc/fb** for systems which contain frame buffer devices looks similar to the following:

0 VESA VGA

E.2.9. /proc/filesystems

This file displays a list of the file system types currently supported by the kernel. Sample output from a generic **/proc/filesystems** file looks similar to the following:

```
nodev sysfs
nodev rootfs
nodev bdev
nodev proc
nodev sockfs
nodev binfmt_misc
nodev usbfs
nodev usbdevfs
nodev futexfs
nodev tmpfs
nodev pipefs
nodev eventpollfs
nodev devpts
ext2
nodev ramfs
nodev hugetlbfs
iso9660
nodev mqueue
ext3
nodev rpc_pipefs
nodev autofs
```

The first column signifies whether the file system is mounted on a block device. Those beginning with **nodev** are not mounted on a device. The second column lists the names of the file systems supported.

The **mount** command cycles through the file systems listed here when one is not specified as an argument.

E.2.10. /proc/interrupts

This file records the number of interrupts per IRQ on the x86 architecture. A standard **/proc/interrupts** looks similar to the following:

```
CPU0
0: 80448940      XT-PIC timer
1: 174412      XT-PIC keyboard
2: 0      XT-PIC cascade
8: 1      XT-PIC rtc
10: 410964      XT-PIC eth0
12: 60330      XT-PIC PS/2 Mouse
14: 1314121     XT-PIC ide0
15: 5195422     XT-PIC ide1
NMI: 0
ERR: 0
```

For a multi-processor machine, this file may look slightly different:

```
CPU0      CPU1
0: 1366814704    0      XT-PIC timer
1: 128      340  IO-APIC-edge keyboard
2: 0      0      XT-PIC cascade
8: 0      1  IO-APIC-edge rtc
12: 5323      5793  IO-APIC-edge PS/2 Mouse
13: 1      0      XT-PIC fpu
16: 11184294  15940594  IO-APIC-level Intel EtherExpress Pro 10/100 Ethernet
20: 8450043  11120093  IO-APIC-level megaraid
30: 10432      10722  IO-APIC-level aic7xxx
31: 23      22  IO-APIC-level aic7xxx
NMI: 0
ERR: 0
```

The first column refers to the IRQ number. Each CPU in the system has its own column and its own number of interrupts per IRQ. The next column reports the type of interrupt, and the last column contains the name of the device that is located at that IRQ.

Each of the types of interrupts seen in this file, which are architecture-specific, mean something different. For x86 machines, the following values are common:

- **XT-PIC** – This is the old AT computer interrupts.
- **IO-APIC-edge** – The voltage signal on this interrupt transitions from low to high, creating an edge, where the interrupt occurs and is only signaled once. This kind of interrupt, as well as the **IO-APIC-level** interrupt, are only seen on systems with processors from the 586 family and higher.
- **IO-APIC-level** – Generates interrupts when its voltage signal is high until the signal is low again.

E.2.11. /proc/iomem

This file shows you the current map of the system's memory for each physical device:

```

00000000-0009fbff : System RAM
0009fc00-0009ffff : reserved
000a0000-000bffff : Video RAM area
000c0000-000c7fff : Video ROM
000f0000-000fffff : System ROM
00100000-07ffffff : System RAM
00100000-00291ba8 : Kernel code
00291ba9-002e09cb : Kernel data
e0000000-e3ffffff : VIA Technologies, Inc. VT82C597 [Apollo VP3] e4000000-e7fffff : PCI Bus #01
e4000000-e4003fff : Matrox Graphics, Inc. MGA G200 AGP
e5000000-e57fffff : Matrox Graphics, Inc. MGA G200 AGP
e8000000-e8ffffff : PCI Bus #01
e8000000-e8ffffff : Matrox Graphics, Inc. MGA G200 AGP
ea000000-ea00007f : Digital Equipment Corporation DECchip 21140 [FasterNet]
ea000000-ea00007f : tulip ffff0000-ffffffff : reserved

```

The first column displays the memory registers used by each of the different types of memory. The second column lists the kind of memory located within those registers and displays which memory registers are used by the kernel within the system RAM or, if the network interface card has multiple Ethernet ports, the memory registers assigned for each port.

E.2.12. /proc/ioports

The output of **/proc/ioports** provides a list of currently registered port regions used for input or output communication with a device. This file can be quite long. The following is a partial listing:

```

0000-001f : dma1
0020-003f : pic1
0040-005f : timer
0060-006f : keyboard
0070-007f : rtc
0080-008f : dma page reg
00a0-00bf : pic2
00c0-00df : dma2
00f0-00ff : fpu
0170-0177 : ide1
01f0-01f7 : ide0
02f8-02ff : serial(auto)
0376-0376 : ide1
03c0-03df : vga+
03f6-03f6 : ide0
03f8-03ff : serial(auto)
0cf8-0cff : PCI conf1
d000-dfff : PCI Bus #01
e000-e00f : VIA Technologies, Inc. Bus Master IDE
e000-e007 : ide0
e008-e00f : ide1
e800-e87f : Digital Equipment Corporation DECchip 21140 [FasterNet]
e800-e87f : tulip

```

The first column gives the I/O port address range reserved for the device listed in the second column.

E.2.13. /proc/kcore

This file represents the physical memory of the system and is stored in the core file format. Unlike most **/proc/** files, **kcore** displays a size. This value is given in bytes and is equal to the size of the physical memory (RAM) used plus 4 KB.

The contents of this file are designed to be examined by a debugger, such as **gdb**, and is not human readable.



WARNING

Do not view the **/proc/kcore** virtual file. The contents of the file scramble text output on the terminal. If this file is accidentally viewed, press **Ctrl+C** to stop the process and then type **reset** to bring back the command line prompt.

E.2.14. /proc/kmsg

This file is used to hold messages generated by the kernel. These messages are then picked up by other programs, such as **/sbin/klogd** or **/bin/dmesg**.

E.2.15. /proc/loadavg

This file provides a look at the load average in regard to both the CPU and IO over time, as well as additional data used by **uptime** and other commands. A sample **/proc/loadavg** file looks similar to the following:

```
0.20 0.18 0.12 1/80 11206
```

The first three columns measure CPU and IO utilization of the last one, five, and 15 minute periods. The fourth column shows the number of currently running processes and the total number of processes. The last column displays the last process ID used.

In addition, load average also refers to the number of processes ready to run (i.e. in the run queue, waiting for a CPU share).

E.2.16. /proc/locks

This file displays the files currently locked by the kernel. The contents of this file contain internal kernel debugging data and can vary tremendously, depending on the use of the system. A sample **/proc/locks** file for a lightly loaded system looks similar to the following:

```
1: POSIX ADVISORY WRITE 3568 fd:00:2531452 0 EOF
2: FLOCK ADVISORY WRITE 3517 fd:00:2531448 0 EOF
3: POSIX ADVISORY WRITE 3452 fd:00:2531442 0 EOF
4: POSIX ADVISORY WRITE 3443 fd:00:2531440 0 EOF
5: POSIX ADVISORY WRITE 3326 fd:00:2531430 0 EOF
6: POSIX ADVISORY WRITE 3175 fd:00:2531425 0 EOF
7: POSIX ADVISORY WRITE 3056 fd:00:2548663 0 EOF
```

Each lock has its own line which starts with a unique number. The second column refers to the class of lock used, with **FLOCK** signifying the older-style UNIX file locks from a **flock** system call and **POSIX** representing the newer POSIX locks from the **lockf** system call.

The third column can have two values: **ADVISORY** or **MANDATORY**. **ADVISORY** means that the lock does not prevent other people from accessing the data; it only prevents other attempts to lock it. **MANDATORY** means that no other access to the data is permitted while the lock is held. The fourth column reveals whether the lock is allowing the holder **READ** or **WRITE** access to the file. The fifth column shows the ID of the process holding the lock. The sixth column shows the ID of the file being locked, in the format of **MAJOR-DEVICE:MINOR-DEVICE:INODE-NUMBER**. The seventh and eighth column shows the start and end of the file's locked region.

E.2.17. /proc/mdstat

This file contains the current information for multiple-disk, RAID configurations. If the system does not contain such a configuration, then **/proc/mdstat** looks similar to the following:

```
Personalities : read_ahead not set unused devices: <none>
```

This file remains in the same state as seen above unless a software RAID or **md** device is present. In that case, view **/proc/mdstat** to find the current status of **mdX** RAID devices.

The **/proc/mdstat** file below shows a system with its **md0** configured as a RAID 1 device, while it is currently re-syncing the disks:

```
Personalities : [linear] [raid1] read_ahead 1024 sectors
md0: active raid1 sda2[1] sdb2[0] 9940 blocks [2/2] [UU] resync=1% finish=12.3min algorithm 2 [3/3]
      [UUU]
unused devices: <none>
```

E.2.18. /proc/meminfo

This is one of the more commonly used files in the **/proc/** directory, as it reports a large amount of valuable information about the system's RAM usage.

The following sample **/proc/meminfo** virtual file is from a system with 2 GB of RAM and 1 GB of swap space:

MemTotal:	1921988 kB
MemFree:	1374408 kB
Buffers:	32688 kB
Cached:	370540 kB
SwapCached:	0 kB
Active:	344604 kB
Inactive:	80800 kB
Active(anon):	22364 kB
Inactive(anon):	4 kB
Active(file):	322240 kB
Inactive(file):	80796 kB
Unevictable:	0 kB
Mlocked:	0 kB
SwapTotal:	1048572 kB
SwapFree:	1048572 kB
Dirty:	48 kB

```

Writeback:          0 kB
AnonPages:        22260 kB
Mapped:           13628 kB
Shmem:            196 kB
Slab:             91648 kB
SReclaimable:    34024 kB
SUnreclaim:      57624 kB
KernelStack:     2880 kB
PageTables:       3620 kB
NFS_Unstable:    0 kB
Bounce:           0 kB
WritebackTmp:     0 kB
CommitLimit:     2009564 kB
Committed_AS:    134216 kB
VmallocTotal:    34359738367 kB
VmallocUsed:     12276 kB
VmallocChunk:   34359712840 kB
HardwareCorrupted: 0 kB
AnonHugePages:   0 kB
HugePages_Total: 0
HugePages_Free:  0
HugePages_Rsvd:  0
HugePages_Surp:  0
Hugepagesize:    2048 kB
DirectMap4k:     8064 kB
DirectMap2M:    2088960 kB

```

While the file shows kilobytes (kB; 1 kB equals 1000 B), it is actually kibibytes (KiB; 1 KiB equals 1024 B). This imprecision in **/proc/meminfo** is known, but is not corrected due to legacy concerns - programs rely on **/proc/meminfo** to specify size with the "kB" string.

Much of the information in **/proc/meminfo** is used by the **free**, **top**, and **ps** commands. In fact, the output of the **free** command is similar in appearance to the contents and structure of **/proc/meminfo**. However, **/proc/meminfo** itself has more details:

- **MemTotal** – Total amount of usable RAM, in kibibytes, which is physical RAM minus a number of reserved bits and the kernel binary code.
- **MemFree** – The amount of physical RAM, in kibibytes, left unused by the system.
- **Buffers** – The amount, in kibibytes, of temporary storage for raw disk blocks.
- **Cached** – The amount of physical RAM, in kibibytes, used as cache memory.
- **SwapCached** – The amount of memory, in kibibytes, that has once been moved into swap, then back into the main memory, but still also remains in the swapfile. This saves I/O, because the memory does not need to be moved into swap again.
- **Active** – The amount of memory, in kibibytes, that has been used more recently and is usually not reclaimed unless absolutely necessary.
- **Inactive** – The amount of memory, in kibibytes, that has been used less recently and is more eligible to be reclaimed for other purposes.
- **Active(anon)** – The amount of anonymous and tmpfs/shmem memory, in kibibytes, that is in active use, or was in active use since the last time the system moved something to swap.

- **Inactive(anon)** – The amount of anonymous and tmpfs/shmem memory, in kibibytes, that is a candidate for eviction.
- **Active(file)** – The amount of file cache memory, in kibibytes, that is in active use, or was in active use since the last time the system reclaimed memory.
- **Inactive(file)** – The amount of file cache memory, in kibibytes, that is newly loaded from the disk, or is a candidate for reclaiming.
- **Unevictable** – The amount of memory, in kibibytes, discovered by the pageout code, that is not evictable because it is locked into memory by user programs.
- **Mlocked** – The total amount of memory, in kibibytes, that is not evictable because it is locked into memory by user programs.
- **SwapTotal** – The total amount of swap available, in kibibytes.
- **SwapFree** – The total amount of swap free, in kibibytes.
- **Dirty** – The total amount of memory, in kibibytes, waiting to be written back to the disk.
- **Writeback** – The total amount of memory, in kibibytes, actively being written back to the disk.
- **AnonPages** – The total amount of memory, in kibibytes, used by pages that are not backed by files and are mapped into userspace page tables.
- **Mapped** – The memory, in kibibytes, used for files that have been mmaped, such as libraries.
- **Shmem** – The total amount of memory, in kibibytes, used by shared memory (shmem) and tmpfs.
- **Slab** – The total amount of memory, in kibibytes, used by the kernel to cache data structures for its own use.
- **SReclaimable** – The part of Slab that can be reclaimed, such as caches.
- **SUnreclaim** – The part of Slab that cannot be reclaimed even when lacking memory.
- **KernelStack** – The amount of memory, in kibibytes, used by the kernel stack allocations done for each task in the system.
- **PageTables** – The total amount of memory, in kibibytes, dedicated to the lowest page table level.
- **NFS_Unstable** – The amount, in kibibytes, of NFS pages sent to the server but not yet committed to the stable storage.
- **Bounce** – The amount of memory, in kibibytes, used for the block device "bounce buffers".
- **WritebackTmp** – The amount of memory, in kibibytes, used by FUSE for temporary writeback buffers.
- **CommitLimit** – The total amount of memory currently available to be allocated on the system based on the overcommit ratio (**vm.overcommit_ratio**). This limit is only adhered to if strict overcommit accounting is enabled (mode 2 in **vm.overcommit_memory**). **CommitLimit** is calculated with the following formula:

$$\frac{([total\ RAM\ pages] - [total\ huge\ TLB\ pages]) * overcommit_ratio}{100} + [total\ swap\ pages]$$

For example, on a system with 1 GB of physical RAM and 7 GB of swap with a `vm.overcommit_ratio` of 30 it would yield a **CommitLimit** of 7.3 GB.

- **Committed_AS** – The total amount of memory, in kibibytes, estimated to complete the workload. This value represents the worst case scenario value, and also includes swap memory.
- **VMallocTotal** – The total amount of memory, in kibibytes, of total allocated virtual address space.
- **VMallocUsed** – The total amount of memory, in kibibytes, of used virtual address space.
- **VMallocChunk** – The largest contiguous block of memory, in kibibytes, of available virtual address space.
- **HardwareCorrupted** – The amount of memory, in kibibytes, with physical memory corruption problems, identified by the hardware and set aside by the kernel so it does not get used.
- **AnonHugePages** – The total amount of memory, in kibibytes, used by huge pages that are not backed by files and are mapped into userspace page tables.
- **HugePages_Total** – The total number of hugepages for the system. The number is derived by dividing **Hugepagesize** by the megabytes set aside for hugepages specified in `/proc/sys/vm/hugetlb_pool`. *This statistic only appears on the x86, Itanium, and AMD64 architectures.*
- **HugePages_Free** – The total number of hugepages available for the system. *This statistic only appears on the x86, Itanium, and AMD64 architectures.*
- **HugePages_Rsvd** – The number of unused huge pages reserved for hugetlbfss.
- **HugePages_Surp** – The number of surplus huge pages.
- **Hugepagesize** – The size for each hugepages unit in kibibytes. By default, the value is 4096 KB on uniprocessor kernels for 32 bit architectures. For SMP, hugemem kernels, and AMD64, the default is 2048 KB. For Itanium architectures, the default is 262144 KB. *This statistic only appears on the x86, Itanium, and AMD64 architectures.*
- **DirectMap4k** – The amount of memory, in kibibytes, mapped into kernel address space with 4 kB page mappings.
- **DirectMap2M** – The amount of memory, in kibibytes, mapped into kernel address space with 2 MB page mappings.

E.2.19. /proc/misc

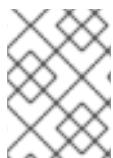
This file lists miscellaneous drivers registered on the miscellaneous major device, which is device number 10:

63	device-mapper	175	apgpart	135	rtc	134	apm_bios
----	---------------	-----	---------	-----	-----	-----	----------

The first column is the minor number of each device, while the second column shows the driver in use.

E.2.20. /proc/modules

This file displays a list of all modules loaded into the kernel. Its contents vary based on the configuration and use of your system, but it should be organized in a similar manner to this sample **/proc/modules** file output:



NOTE

This example has been reformatted into a readable format. Most of this information can also be viewed via the **/sbin/lsmod** command.

```
nfs      170109  0 -      Live 0x129b0000
lockd    51593   1 nfs,     Live 0x128b0000
nls_utf8 1729   0 -      Live 0x12830000
vfat     12097   0 -      Live 0x12823000
fat      38881   1 vfat,   Live 0x1287b000
autofs4  20293   2 -      Live 0x1284f000
sunrpc   140453  3 nfs,lockd, Live 0x12954000
3c59x    33257   0 -      Live 0x12871000
uhci_hcd 28377   0 -      Live 0x12869000
md5      3777    1 -      Live 0x1282c000
ipv6     211845  16 -     Live 0x128de000
ext3     92585   2 -      Live 0x12886000
jbd      65625   1 ext3,   Live 0x12857000
dm_mod   46677   3 -      Live 0x12833000
```

The first column contains the name of the module.

The second column refers to the memory size of the module, in bytes.

The third column lists how many instances of the module are currently loaded. A value of zero represents an unloaded module.

The fourth column states if the module depends upon another module to be present in order to function, and lists those other modules.

The fifth column lists what load state the module is in: **Live**, **Loading**, or **Unloading** are the only possible values.

The sixth column lists the current kernel memory offset for the loaded module. This information can be useful for debugging purposes, or for profiling tools such as **oprofile**.

E.2.21. /proc/mounts

This file provides a list of all mounts in use by the system:

```
rootfs / rootfs rw 0 0
/proc /proc proc rw,nodiratime 0 0 none
/dev ramfs rw 0 0
/dev/mapper/VolGroup00-LogVol00 / ext3 rw 0 0
none /dev ramfs rw 0 0
/proc /proc proc rw,nodiratime 0 0
/sys /sys sysfs rw 0 0
none /dev/pts devpts rw 0 0
```

```
usbdevfs /proc/bus/usb usbdevfs rw 0 0
/dev/hda1 /boot ext3 rw 0 0
none /dev/shm tmpfs rw 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
sunrpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw 0 0
```

The output found here is similar to the contents of **/etc/mtab**, except that **/proc/mounts** is more up-to-date.

The first column specifies the device that is mounted, the second column reveals the mount point, and the third column tells the file system type, and the fourth column tells you if it is mounted read-only (**ro**) or read-write (**rw**). The fifth and sixth columns are dummy values designed to match the format used in **/etc/mtab**.

E.2.22. **/proc/mtrr**

This file refers to the current Memory Type Range Registers (MTRRs) in use with the system. If the system architecture supports MTRRs, then the **/proc/mtrr** file may look similar to the following:

```
reg00: base=0x00000000 ( 0MB), size= 256MB: write-back, count=1
reg01: base=0xe8000000 (3712MB), size= 32MB: write-combining, count=1
```

MTRRs are used with the Intel P6 family of processors (Pentium II and higher) and control processor access to memory ranges. When using a video card on a PCI or AGP bus, a properly configured **/proc/mtrr** file can increase performance more than 150%.

Most of the time, this value is properly configured by default. More information on manually configuring this file can be found locally at the following location:

```
/usr/share/doc/kernel-doc-<kernel_version>/Documentation/<arch>/mtrr.txt
```

E.2.23. **/proc/partitions**

This file contains partition block allocation information. A sampling of this file from a basic system looks similar to the following:

```
major minor #blocks name
 3      0   19531250 hda
 3      1    104391 hda1
 3      2   19422585 hda2
253     0   22708224 dm-0
253     1    524288 dm-1
```

Most of the information here is of little importance to the user, except for the following columns:

- **major** – The major number of the device with this partition. The major number in the **/proc/partitions**, (3), corresponds with the block device **ide0**, in **/proc/devices**.
- **minor** – The minor number of the device with this partition. This serves to separate the partitions into different physical devices and relates to the number at the end of the name of the partition.
- **#blocks** – Lists the number of physical disk blocks contained in a particular partition.

- **name** – The name of the partition.

E.2.24. /proc/slabinfo

This file gives full information about memory usage on the *slab* level. Linux kernels greater than version 2.2 use *slab pools* to manage memory above the page level. Commonly used objects have their own slab pools.

Instead of parsing the highly verbose **/proc/slabinfo** file manually, the **/usr/bin/slabtop** program displays kernel slab cache information in real time. This program allows for custom configurations, including column sorting and screen refreshing.

A sample screen shot of **/usr/bin/slabtop** usually looks like the following example:

OBJS	ACTIVE	USE	OBJ	SIZE	SLABS	OBJ/SLAB	CACHE	SIZE	NAME
44814	43159	96%	0.62K	7469	6	29876K	ext3_inode_cache		
36900	34614	93%	0.05K	492	75	1968K	buffer_head		
35213	33124	94%	0.16K	1531	23	6124K	dentry_cache		
7364	6463	87%	0.27K	526	14	2104K	radix_tree_node		
2585	1781	68%	0.08K	55	47	220K	vm_area_struct		
2263	2116	93%	0.12K	73	31	292K	size-128		
1904	1125	59%	0.03K	16	119	64K	size-32		
1666	768	46%	0.03K	14	119	56K	anon_vma		
1512	1482	98%	0.44K	168	9	672K	inode_cache		
1464	1040	71%	0.06K	24	61	96K	size-64		
1320	820	62%	0.19K	66	20	264K	filp		
678	587	86%	0.02K	3	226	12K	dm_io		
678	587	86%	0.02K	3	226	12K	dm_tio		
576	574	99%	0.47K	72	8	288K	proc_inode_cache		
528	514	97%	0.50K	66	8	264K	size-512		
492	372	75%	0.09K	12	41	48K	bio		
465	314	67%	0.25K	31	15	124K	size-256		
452	331	73%	0.02K	2	226	8K	biovec-1		
420	420	100%	0.19K	21	20	84K	skbuff_head_cache		
305	256	83%	0.06K	5	61	20K	biovec-4		
290	4	1%	0.01K	1	290	4K	revoke_table		
264	264	100%	4.00K	264	1	1056K	size-4096		
260	256	98%	0.19K	13	20	52K	biovec-16		
260	256	98%	0.75K	52	5	208K	biovec-64		

Some of the more commonly used statistics in **/proc/slabinfo** that are included into **/usr/bin/slabtop** include:

- **OBJS** – The total number of objects (memory blocks), including those in use (allocated), and some spares not in use.
- **ACTIVE** – The number of objects (memory blocks) that are in use (allocated).
- **USE** – Percentage of total objects that are active. ((ACTIVE/OBJS)(100))
- **OBJ SIZE** – The size of the objects.

- **SLABS** – The total number of slabs.
- **OBJ/SLAB** – The number of objects that fit into a slab.
- **CACHE SIZE** – The cache size of the slab.
- **NAME** – The name of the slab.

For more information on the **/usr/bin/slabtop** program, refer to the **slabtop** man page.

E.2.25. /proc/stat

This file keeps track of a variety of different statistics about the system since it was last restarted. The contents of **/proc/stat**, which can be quite long, usually begins like the following example:

```
cpu 259246 7001 60190 34250993 137517 772 0
cpu0 259246 7001 60190 34250993 137517 772 0
intr 354133732 347209999 2272 0 4 4 0 0 3 1 1249247 0 0 80143 0 422626 5169433
ctxt 12547729
bttime 1093631447
processes 130523
procs_running 1
procs_blocked 0
preempt 5651840
cpu 209841 1554 21720 118519346 72939 154 27168
cpu0 42536 798 4841 14790880 14778 124 3117
cpu1 24184 569 3875 14794524 30209 29 3130
cpu2 28616 11 2182 14818198 4020 1 3493
cpu3 35350 6 2942 14811519 3045 0 3659
cpu4 18209 135 2263 14820076 12465 0 3373
cpu5 20795 35 1866 14825701 4508 0 3615
cpu6 21607 0 2201 14827053 2325 0 3334
cpu7 18544 0 1550 14831395 1589 0 3447
intr 15239682 14857833 6 0 6 6 0 5 0 1 0 0 0 29 0 2 0 0 0 0 0 0 0 94982 0 286812
ctxt 4209609
bttime 1078711415
processes 21905
procs_running 1
procs_blocked 0
```

Some of the more commonly used statistics include:

- **cpu** – Measures the number of *jiffies* (1/100 of a second for x86 systems) that the system has been in user mode, user mode with low priority (nice), system mode, idle task, I/O wait, IRQ (hardirq), and softirq respectively. The IRQ (hardirq) is the direct response to a hardware event. The IRQ takes minimal work for queuing the "heavy" work up for the softirq to execute. The softirq runs at a lower priority than the IRQ and therefore may be interrupted more frequently. The total for all CPUs is given at the top, while each individual CPU is listed below with its own statistics. The following example is a 4-way Intel Pentium Xeon configuration with multi-threading enabled, therefore showing four physical processors and four virtual processors totaling eight processors.
- **page** – The number of memory pages the system has written in and out to disk.
- **swap** – The number of swap pages the system has brought in and out.

- **intr** – The number of interrupts the system has experienced.
- **btime** – The boot time, measured in the number of seconds since January 1, 1970, otherwise known as the epoch.

E.2.26. /proc/swaps

This file measures swap space and its utilization. For a system with only one swap partition, the output of **/proc/swaps** may look similar to the following:

Filename	Type	Size	Used	Priority
/dev/mapper/VolGroup00-LogVol01	partition	524280	0	-1

While some of this information can be found in other files in the **/proc/** directory, **/proc/swap** provides a snapshot of every swap file name, the type of swap space, the total size, and the amount of space in use (in kilobytes). The priority column is useful when multiple swap files are in use. The lower the priority, the more likely the swap file is to be used.

E.2.27. /proc/sysrq-trigger

Using the **echo** command to write to this file, a remote root user can execute most System Request Key commands remotely as if at the local terminal. To **echo** values to this file, the **/proc/sys/kernel/sysrq** must be set to a value other than **0**. For more information about the System Request Key, see [Section E.3.9.3, "/proc/sys/kernel/"](#).

Although it is possible to write to this file, it cannot be read, even by the root user.

E.2.28. /proc/uptime

This file contains information detailing how long the system has been on since its last restart. The output of **/proc/uptime** is quite minimal:

350735.47 234388.90

The first value represents the total number of seconds the system has been up. The second value is the sum of how much time each core has spent idle, in seconds. Consequently, the second value may be greater than the overall system uptime on systems with multiple cores.

E.2.29. /proc/version

This file specifies the version of the Linux kernel, the version of **gcc** used to compile the kernel, and the time of kernel compilation. It also contains the kernel compiler's user name (in parentheses).

Linux version 2.6.8-1.523 (user@foo.redhat.com) (gcc version 3.4.1 20040714 \ (Red Hat Enterprise Linux 3.4.1-7)) #1 Mon Aug 16 13:27:03 EDT 2004

This information is used for a variety of purposes, including the version data presented when a user logs in.

E.3. DIRECTORIES WITHIN /PROC/

Common groups of information concerning the kernel are grouped into directories and subdirectories within the **/proc/** directory.

E.3.1. Process Directories

Every **/proc/** directory contains a number of directories with numerical names. A listing of them may be similar to the following:

```
dr-xr-xr-x 3 root  root      0 Feb 13 01:28 1
dr-xr-xr-x 3 root  root      0 Feb 13 01:28 1010
dr-xr-xr-x 3 xfs   xfs       0 Feb 13 01:28 1087
dr-xr-xr-x 3 daemon  daemon   0 Feb 13 01:28 1123
dr-xr-xr-x 3 root   root     0 Feb 13 01:28 11307
dr-xr-xr-x 3 apache  apache   0 Feb 13 01:28 13660
dr-xr-xr-x 3 rpc    rpc      0 Feb 13 01:28 637
dr-xr-xr-x 3 rpcuser rpcuser  0 Feb 13 01:28 666
```

These directories are called *process directories*, as they are named after a program's process ID and contain information specific to that process. The owner and group of each process directory is set to the user running the process. When the process is terminated, its **/proc/** process directory vanishes.

Each process directory contains the following files:

- **cmdline** – Contains the command issued when starting the process.
- **cwd** – A symbolic link to the current working directory for the process.
- **environ** – A list of the environment variables for the process. The environment variable is given in all upper-case characters, and the value is in lower-case characters.
- **exe** – A symbolic link to the executable of this process.
- **fd** – A directory containing all of the file descriptors for a particular process. These are given in numbered links:

```
total 0
lrwx----- 1 root  root      64 May  8 11:31 0 -> /dev/null
lrwx----- 1 root  root      64 May  8 11:31 1 -> /dev/null
lrwx----- 1 root  root      64 May  8 11:31 2 -> /dev/null
lrwx----- 1 root  root      64 May  8 11:31 3 -> /dev/ptmx
lrwx----- 1 root  root      64 May  8 11:31 4 -> socket:[7774817]
lrwx----- 1 root  root      64 May  8 11:31 5 -> /dev/ptmx
lrwx----- 1 root  root      64 May  8 11:31 6 -> socket:[7774829]
lrwx----- 1 root  root      64 May  8 11:31 7 -> /dev/ptmx
```

- **maps** – A list of memory maps to the various executables and library files associated with this process. This file can be rather long, depending upon the complexity of the process, but sample output from the **sshd** process begins like the following:

```
08048000-08086000 r-xp 00000000 03:03 391479  /usr/sbin/sshd
08086000-08088000 rw-p 0003e000 03:03 391479 /usr/sbin/sshd
08088000-08095000 rwxp 00000000 00:00 0
40000000-40013000 r-xp 00000000 03:03 293205 /lib/ld-2.2.5.so
40013000-40014000 rw-p 00013000 03:03 293205 /lib/ld-2.2.5.so
40031000-40038000 r-xp 00000000 03:03 293282 /lib/libpam.so.0.75
40038000-40039000 rw-p 00006000 03:03 293282 /lib/libpam.so.0.75
```

```
40039000-4003a000 rw-p 00000000 00:00 0
4003a000-4003c000 r-xp 00000000 03:03 293218 /lib/libdl-2.2.5.so
4003c000-4003d000 rw-p 00001000 03:03 293218 /lib/libdl-2.2.5.so
```

- **mem** – The memory held by the process. This file cannot be read by the user.
- **root** – A link to the root directory of the process.
- **stat** – The status of the process.
- **statm** – The status of the memory in use by the process. Below is a sample **/proc/statm** file:

```
263 210 210 5 0 205 0
```

The seven columns relate to different memory statistics for the process. From left to right, they report the following aspects of the memory used:

1. Total program size, in kilobytes.
2. Size of memory portions, in kilobytes.
3. Number of pages that are shared.
4. Number of pages that are code.
5. Number of pages of data/stack.
6. Number of library pages.
7. Number of dirty pages.

- **status** – The status of the process in a more readable form than **stat** or **statm**. Sample output for **sshd** looks similar to the following:

```
Name: sshd
State: S (sleeping)
Tgid: 797
Pid: 797
PPid: 1
TracerPid: 0
Uid: 0 0 0 0
Gid: 0 0 0 0
FDSize: 32
Groups:
VmSize: 3072 kB
VmLck: 0 kB
VmRSS: 840 kB
VmData: 104 kB
VmStk: 12 kB
VmExe: 300 kB
VmLib: 2528 kB
SigPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 8000000000001000
SigCgt: 0000000000014005
```

```
CapInh: 0000000000000000
CapPrm: 00000000fffffeff
CapEff: 00000000fffffeff
```

The information in this output includes the process name and ID, the state (such as **S (sleeping)** or **R (running)**), user/group ID running the process, and detailed data regarding memory usage.

E.3.1.1. /proc/self/

The **/proc/self/** directory is a link to the currently running process. This allows a process to look at itself without having to know its process ID.

Within a shell environment, a listing of the **/proc/self/** directory produces the same contents as listing the process directory for that process.

E.3.2. /proc/bus/

This directory contains information specific to the various buses available on the system. For example, on a standard system containing PCI and USB buses, current data on each of these buses is available within a subdirectory within **/proc/bus/** by the same name, such as **/proc/bus/pci/**.

The subdirectories and files available within **/proc/bus/** vary depending on the devices connected to the system. However, each bus type has at least one directory. Within these bus directories are normally at least one subdirectory with a numerical name, such as **001**, which contain binary files.

For example, the **/proc/bus/usb/** subdirectory contains files that track the various devices on any USB buses, as well as the drivers required for them. The following is a sample listing of a **/proc/bus/usb/** directory:

```
total 0 dr-xr-xr-x  1 root    root      0 May  3 16:25 001
-r--r--r--  1 root    root      0 May  3 16:25 devices
-r--r--r--  1 root    root      0 May  3 16:25 drivers
```

The **/proc/bus/usb/001/** directory contains all devices on the first USB bus and the **devices** file identifies the USB root hub on the motherboard.

The following is a example of a **/proc/bus/usb/devices** file:

```
T: Bus=01 Lev=00 Prnt=00 Port=00 Cnt=00 Dev#= 1 Spd=12 MxCh= 2
B: Alloc= 0/900 us ( 0%), #Int= 0, #Iso= 0
D: Ver= 1.00 Cls=09(hub ) Sub=00 Prot=00 MxPS= 8 #Cfgs= 1
P: Vendor=0000 ProdID=0000 Rev= 0.00
S: Product=USB UHCI Root Hub
S: SerialNumber=d400
C:* #Ifs= 1 Cfg#= 1 Atr=40 MxPwr= 0mA
I: If#= 0 Alt= 0 #EPs= 1 Cls=09(hub ) Sub=00 Prot=00 Driver=hub
E: Ad=81(I) Atr=03(Int.) MxPS= 8 Lvl=255ms
```

E.3.3. /proc/bus/pci

Later versions of the 2.6 Linux kernel have obsoleted the **/proc/pci** directory in favor of the **/proc/bus/pci** directory. Although you can get a list of all PCI devices present on the system using the command **cat /proc/bus/pci/devices**, the output is difficult to read and interpret.

For a human-readable list of PCI devices, run the following command:

```
~]# /sbin/lspci -vb
00:00.0 Host bridge: Intel Corporation 82X38/X48 Express DRAM Controller
    Subsystem: Hewlett-Packard Company Device 1308
    Flags: bus master, fast devsel, latency 0
    Capabilities: [e0] Vendor Specific Information <?>
    Kernel driver in use: x38_edac
    Kernel modules: x38_edac

00:01.0 PCI bridge: Intel Corporation 82X38/X48 Express Host-Primary PCI Express Bridge (prog-if
00 [Normal decode])
    Flags: bus master, fast devsel, latency 0
    Bus: primary=00, secondary=01, subordinate=01, sec-latency=0
    I/O behind bridge: 00001000-000001fff
    Memory behind bridge: f0000000-f2fffff
    Capabilities: [88] Subsystem: Hewlett-Packard Company Device 1308
    Capabilities: [80] Power Management version 3
    Capabilities: [90] MSI: Enable+ Count=1/1 Maskable- 64bit-
    Capabilities: [a0] Express Root Port (Slot+), MSI 00
    Capabilities: [100] Virtual Channel <?>
    Capabilities: [140] Root Complex Link <?>
    Kernel driver in use: pcieport
    Kernel modules: shpchp

00:1a.0 USB Controller: Intel Corporation 82801I (ICH9 Family) USB UHCI Controller #4 (rev 02)
(prog-if 00 [UHCI])
    Subsystem: Hewlett-Packard Company Device 1308
    Flags: bus master, medium devsel, latency 0, IRQ 5
    I/O ports at 2100
    Capabilities: [50] PCI Advanced Features
    Kernel driver in use: uhci_hcd
[output truncated]
```

The output is a sorted list of all IRQ numbers and addresses as seen by the cards on the PCI bus instead of as seen by the kernel. Beyond providing the name and version of the device, this list also gives detailed IRQ information so an administrator can quickly look for conflicts.

E.3.4. /proc/driver/

This directory contains information for specific drivers in use by the kernel.

A common file found here is **rtc** which provides output from the driver for the system's *Real Time Clock (RTC)*, the device that keeps the time while the system is switched off. Sample output from **/proc/driver/rtc** looks like the following:

```
rtc_time      : 16:21:00
rtc_date      : 2004-08-31
rtc_epoch     : 1900
alarm         : 21:16:27
DST_enable    : no
BCD          : yes
24hr         : yes
square_wave   : no
alarm_IRQ     : no
```

```
update_IRQ      : no
periodic_IRQ   : no
periodic_freq  : 1024
batt_status    : okay
```

For more information about the RTC, see the following installed documentation:

`/usr/share/doc/kernel-doc-<kernel_version>/Documentation/rtc.txt`.

E.3.5. /proc/fs

This directory shows which file systems are exported. If running an NFS server, typing `cat /proc/fs/nfsd(exports)` displays the file systems being shared and the permissions granted for those file systems. For more on file system sharing with NFS, see the *Network File System (NFS)* chapter of the *Storage Administration Guide*.

E.3.6. /proc/irq/

This directory is used to set IRQ to CPU affinity, which allows the system to connect a particular IRQ to only one CPU. Alternatively, it can exclude a CPU from handling any IRQs.

Each IRQ has its own directory, allowing for the individual configuration of each IRQ. The `/proc/irq/prof_cpu_mask` file is a bitmask that contains the default values for the `smp_affinity` file in the IRQ directory. The values in `smp_affinity` specify which CPUs handle that particular IRQ.

For more information about the `/proc/irq/` directory, see the following installed documentation:

`/usr/share/doc/kernel-doc-kernel_version/Documentation/filesystems/proc.txt`

E.3.7. /proc/net/

This directory provides a comprehensive look at various networking parameters and statistics. Each directory and virtual file within this directory describes aspects of the system's network configuration. Below is a partial list of the `/proc/net/` directory:

- **arp** – Lists the kernel's ARP table. This file is particularly useful for connecting a hardware address to an IP address on a system.
- **atm/** directory – The files within this directory contain *Asynchronous Transfer Mode (ATM)* settings and statistics. This directory is primarily used with ATM networking and ADSL cards.
- **dev** – Lists the various network devices configured on the system, complete with transmit and receive statistics. This file displays the number of bytes each interface has sent and received, the number of packets inbound and outbound, the number of errors seen, the number of packets dropped, and more.
- **dev_mcast** – Lists Layer2 multicast groups on which each device is listening.
- **igmp** – Lists the IP multicast addresses which this system joined.
- **ip_conntrack** – Lists tracked network connections for machines that are forwarding IP connections.
- **ip_tables_names** – Lists the types of `iptables` in use. This file is only present if `iptables` is active on the system and contains one or more of the following values: `filter`, `mangle`, or `nat`.

- **ip_mr_cache** – Lists the multicast routing cache.
- **ip_mr_vif** – Lists multicast virtual interfaces.
- **netstat** – Contains a broad yet detailed collection of networking statistics, including TCP timeouts, SYN cookies sent and received, and much more.
- **psched** – Lists global packet scheduler parameters.
- **raw** – Lists raw device statistics.
- **route** – Lists the kernel's routing table.
- **rt_cache** – Contains the current routing cache.
- **snmp** – List of Simple Network Management Protocol (SNMP) data for various networking protocols in use.
- **sockstat** – Provides socket statistics.
- **tcp** – Contains detailed TCP socket information.
- **tr_rif** – Lists the token ring RIF routing table.
- **udp** – Contains detailed UDP socket information.
- **unix** – Lists UNIX domain sockets currently in use.
- **wireless** – Lists wireless interface data.

E.3.8. /proc/scsi/

The primary file in this directory is **/proc/scsi/scsi**, which contains a list of every recognized SCSI device. From this listing, the type of device, as well as the model name, vendor, SCSI channel and ID data is available.

For example, if a system contains a SCSI CD-ROM, a tape drive, a hard drive, and a RAID controller, this file looks similar to the following:

```
Attached devices:
Host: scsi1
Channel: 00
Id: 05
Lun: 00
Vendor: NEC
Model: CD-ROM DRIVE:466
Rev: 1.06
Type: CD-ROM
ANSI SCSI revision: 02
Host: scsi1
Channel: 00
Id: 06
Lun: 00
Vendor: ARCHIVE
Model: Python 04106-XXX
Rev: 7350
```

```
Type: Sequential-Access
ANSI SCSI revision: 02
Host: scsi2
Channel: 00
Id: 06
Lun: 00
Vendor: DELL
Model: 1x6 U2W SCSI BP
Rev: 5.35
Type: Processor
ANSI SCSI revision: 02
Host: scsi2
Channel: 02
Id: 00
Lun: 00
Vendor: MegaRAID
Model: LD0 RAID5 34556R
Rev: 1.01
Type: Direct-Access
ANSI SCSI revision: 02
```

Each SCSI driver used by the system has its own directory within **/proc/scsi/**, which contains files specific to each SCSI controller using that driver. From the previous example, **aic7xxx/** and **megaraid/** directories are present, since two drivers are in use. The files in each of the directories typically contain an I/O address range, IRQ information, and statistics for the SCSI controller using that driver. Each controller can report a different type and amount of information. The Adaptec AIC-7880 Ultra SCSI host adapter's file in this example system produces the following output:

```
Adaptec AIC7xxx driver version: 5.1.20/3.2.4
Compile Options:
TCQ Enabled By Default : Disabled
AIC7XXX_PROC_STATS    : Enabled
AIC7XXX_RESET_DELAY   : 5
Adapter Configuration:
SCSI Adapter: Adaptec AIC-7880 Ultra SCSI host adapter
Ultra Narrow Controller  PCI MMAPed
I/O Base: 0xfcffe000
Adapter SEEPROM Config: SEEPROM found and used.
Adaptec SCSI BIOS: Enabled
IRQ: 30
SCBs: Active 0, Max Active 1, Allocated 15, HW 16, Page 255
Interrupts: 33726
BIOS Control Word: 0x18a6
Adapter Control Word: 0x1c5f
Extended Translation: Enabled
Disconnect Enable Flags: 0x00ff
Ultra Enable Flags: 0x0020
Tag Queue Enable Flags: 0x0000
Ordered Queue Tag Flags: 0x0000
Default Tag Queue Depth: 8
Tagged Queue By Device array for aic7xxx
host instance 1: {255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255}
Actual queue depth per device for aic7xxx host instance 1: {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}
Statistics:

(scsi1:0:5:0) Device using Narrow/Sync transfers at 20.0 MByte/sec, offset 15
```

Transinfo settings: current(12/15/0/0), goal(12/15/0/0), user(12/15/0/0)

Total transfers 0 (0 reads and 0 writes)

< 2K	2K+	4K+	8K+	16K+	32K+	64K+	128K+
Reads:	0	0	0	0	0	0	0
Writes:	0	0	0	0	0	0	0

(scsi1:0:6:0) Device using Narrow/Sync transfers at 10.0 MByte/sec, offset 15

Transinfo settings: current(25/15/0/0), goal(12/15/0/0), user(12/15/0/0)

Total transfers 132 (0 reads and 132 writes)

< 2K	2K+	4K+	8K+	16K+	32K+	64K+	128K+
Reads:	0	0	0	0	0	0	0
Writes:	0	0	0	1	131	0	0

This output reveals the transfer speed to the SCSI devices connected to the controller based on channel ID, as well as detailed statistics concerning the amount and sizes of files read or written by that device. For example, this controller is communicating with the CD-ROM at 20 megabytes per second, while the tape drive is only communicating at 10 megabytes per second.

E.3.9. /proc/sys/

The **/proc/sys/** directory is different from others in **/proc/** because it not only provides information about the system but also allows the system administrator to immediately enable and disable kernel features.



WARNING

Use caution when changing settings on a production system using the various files in the **/proc/sys/** directory. Changing the wrong setting may render the kernel unstable, requiring a system reboot.

For this reason, be sure the options are valid for that file before attempting to change any value in **/proc/sys/**.

A good way to determine if a particular file can be configured, or if it is only designed to provide information, is to list it with the **-l** option at the shell prompt. If the file is writable, it may be used to configure the kernel. For example, a partial listing of **/proc/sys/fs** looks like the following:

```
-r--r--r-- 1 root root      0 May 10 16:14 dentry-state
-rw-r--r-- 1 root root      0 May 10 16:14 dir-notify-enable
-rw-r--r-- 1 root root      0 May 10 16:14 file-max
-r--r--r-- 1 root root      0 May 10 16:14 file-nr
```

In this listing, the files **dir-notify-enable** and **file-max** can be written to and, therefore, can be used to configure the kernel. The other files only provide feedback on current settings.

Changing a value within a **/proc/sys/** file is done by echoing the new value into the file. For example, to enable the System Request Key on a running kernel, type the command:

```
echo 1 > /proc/sys/kernel/sysrq
```

This changes the value for **sysrq** from **0** (off) to **1** (on).

A few **/proc/sys/** configuration files contain more than one value. To correctly send new values to them, place a space character between each value passed with the **echo** command, such as is done in this example:

```
echo 4 2 45 > /proc/sys/kernel/acct
```



NOTE

Any configuration changes made using the **echo** command disappear when the system is restarted. To make configuration changes take effect after the system is rebooted, see [Section E.4, "Using the sysctl Command"](#).

The **/proc/sys/** directory contains several subdirectories controlling different aspects of a running kernel.

E.3.9.1. **/proc/sys/dev/**

This directory provides parameters for particular devices on the system. Most systems have at least two directories, **cdrom/** and **raid/**. Customized kernels can have other directories, such as **parport/**, which provides the ability to share one parallel port between multiple device drivers.

The **cdrom/** directory contains a file called **info**, which reveals a number of important CD-ROM parameters:

```
CD-ROM information, Id: cdrom.c 3.20 2003/12/17
drive name:      hdc
drive speed:    48
drive # of slots: 1
Can close tray:  1
Can open tray:   1
Can lock tray:   1
Can change speed: 1
Can select disk: 0
Can read multisession: 1
Can read MCN:    1
Reports media changed: 1
Can play audio:  1
Can write CD-R:   0
Can write CD-RW:  0
Can read DVD:    0
Can write DVD-R:  0
Can write DVD-RAM: 0
Can read MRW:    0
Can write MRW:   0
Can write RAM:   0
```

This file can be quickly scanned to discover the qualities of an unknown CD-ROM. If multiple CD-ROMs are available on a system, each device is given its own column of information.

Various files in **/proc/sys/dev/cdrom**, such as **autoclose** and **checkmedia**, can be used to control the system's CD-ROM. Use the **echo** command to enable or disable these features.

If RAID support is compiled into the kernel, a `/proc/sys/dev/raid/` directory becomes available with at least two files in it: **speed_limit_min** and **speed_limit_max**. These settings determine the acceleration of RAID devices for I/O intensive tasks, such as resyncing the disks.

E.3.9.2. `/proc/sys/fs/`

This directory contains an array of options and information concerning various aspects of the file system, including quota, file handle, inode, and dentry information.

The `bifmt_misc/` directory is used to provide kernel support for miscellaneous binary formats.

The important files in `/proc/sys/fs/` include:

- **dentry-state** – Provides the status of the directory cache. The file looks similar to the following:

```
57411 52939 45 0 0 0
```

The first number reveals the total number of directory cache entries, while the second number displays the number of unused entries. The third number tells the number of seconds between when a directory has been freed and when it can be reclaimed, and the fourth measures the pages currently requested by the system. The last two numbers are not used and display only zeros.

- **file-max** – Lists the maximum number of file handles that the kernel allocates. Raising the value in this file can resolve errors caused by a lack of available file handles.
- **file-nr** – Lists the number of allocated file handles, used file handles, and the maximum number of file handles.
- **overflowgid** and **overflowuid** – Defines the fixed group ID and user ID, respectively, for use with file systems that only support 16-bit group and user IDs.

E.3.9.3. `/proc/sys/kernel/`

This directory contains a variety of different configuration files that directly affect the operation of the kernel. Some of the most important files include:

- **acct** – Controls the suspension of process accounting based on the percentage of free space available on the file system containing the log. By default, the file looks like the following:

```
4 2 30
```

The first value dictates the percentage of free space required for logging to resume, while the second value sets the threshold percentage of free space when logging is suspended. The third value sets the interval, in seconds, that the kernel polls the file system to see if logging should be suspended or resumed.

- **ctrl-alt-del** – Controls whether **Ctrl+Alt+Delete** gracefully restarts the computer using **init (0)** or forces an immediate reboot without syncing the dirty buffers to disk **(1)**.
- **domainname** – Configures the system domain name, such as **example.com**.
- **exec-shield** – Configures the Exec Shield feature of the kernel. Exec Shield provides protection against certain types of buffer overflow attacks.

There are two possible values for this virtual file:

- **0** – Disables Exec Shield.
- **1** – Enables Exec Shield. This is the default value.



IMPORTANT

If a system is running security-sensitive applications that were started while Exec Shield was disabled, these applications must be restarted when Exec Shield is enabled in order for Exec Shield to take effect.

- **hostname** – Configures the system host name, such as **www.example.com**.
- **hotplug** – Configures the utility to be used when a configuration change is detected by the system. This is primarily used with USB and Cardbus PCI. The default value of **/sbin/hotplug** should not be changed unless testing a new program to fulfill this role.
- **modprobe** – Sets the location of the program used to load kernel modules. The default value is **/sbin/modprobe** which means **kmod** calls it to load the module when a kernel thread calls **kmod**.
- **msgmax** – Sets the maximum size of any message sent from one process to another and is set to **8192** bytes by default. Be careful when raising this value, as queued messages between processes are stored in non-swappable kernel memory. Any increase in **msgmax** would increase RAM requirements for the system.
- **msgmnb** – Sets the maximum number of bytes in a single message queue. The default is **16384**.
- **msgmni** – Sets the maximum number of message queue identifiers. The default is **4008**.
- **osrelease** – Lists the Linux kernel release number. This file can only be altered by changing the kernel source and recompiling.
- **ostype** – Displays the type of operating system. By default, this file is set to **Linux**, and this value can only be changed by changing the kernel source and recompiling.
- **overflowgid** and **overflowuid** – Defines the fixed group ID and user ID, respectively, for use with system calls on architectures that only support 16-bit group and user IDs.
- **panic** – Defines the number of seconds the kernel postpones rebooting when the system experiences a kernel panic. By default, the value is set to **0**, which disables automatic rebooting after a panic.
- **printk** – This file controls a variety of settings related to printing or logging error messages. Each error message reported by the kernel has a *loglevel* associated with it that defines the importance of the message. The loglevel values break down in this order:
 - **0** – Kernel emergency. The system is unusable.
 - **1** – Kernel alert. Action must be taken immediately.
 - **2** – Condition of the kernel is considered critical.
 - **3** – General kernel error condition.
 - **4** – General kernel warning condition.

- 5 – Kernel notice of a normal but significant condition.
- 6 – Kernel informational message.
- 7 – Kernel debug-level messages.

Four values are found in the **printk** file:

```
6 4 1 7
```

Each of these values defines a different rule for dealing with error messages. The first value, called the *console loglevel*, defines the lowest priority of messages printed to the console. (Note that, the lower the priority, the higher the loglevel number.) The second value sets the default loglevel for messages without an explicit loglevel attached to them. The third value sets the lowest possible loglevel configuration for the console loglevel. The last value sets the default value for the console loglevel.

- **random**/ directory – Lists a number of values related to generating random numbers for the kernel.
- **sem** – Configures semaphore settings within the kernel. A semaphore is a System V IPC object that is used to control utilization of a particular process.
- **shmall** – Sets the total amount of shared memory that can be used at one time on the system, in bytes. By default, this value is **2097152**.
- **shmmmax** – Sets the largest shared memory segment size allowed by the kernel. By default, this value is **33554432**. However, the kernel supports much larger values than this.
- **shmmni** – Sets the maximum number of shared memory segments for the whole system. By default, this value is **4096**.
- **sysrq** – Activates the System Request Key, if this value is set to anything other than zero (**0**), the default.

The System Request Key allows immediate input to the kernel through simple key combinations. For example, the System Request Key can be used to immediately shut down or restart a system, sync all mounted file systems, or dump important information to the console. To initiate a System Request Key, type **Alt+SysRq+system request code**. Replace *system request code* with one of the following system request codes:

- **r** – Disables raw mode for the keyboard and sets it to XLATE (a limited keyboard mode which does not recognize modifiers such as **Alt**, **Ctrl**, or **Shift** for all keys).
- **k** – Kills all processes active in a virtual console. Also called Secure Access Key (SAK), it is often used to verify that the login prompt is spawned from **init** and not a trojan copy designed to capture user names and passwords.
- **b** – Reboots the kernel without first unmounting file systems or syncing disks attached to the system.
- **c** – Crashes the system without first unmounting file systems or syncing disks attached to the system.
- **o** – Shuts off the system.
- **s** – Attempts to sync disks attached to the system.

- **u** – Attempts to unmount and remount all file systems as read-only.
- **p** – Outputs all flags and registers to the console.
- **t** – Outputs a list of processes to the console.
- **m** – Outputs memory statistics to the console.
- **0** through **9** – Sets the log level for the console.
- **e** – Kills all processes except **init** using SIGTERM.
- **i** – Kills all processes except **init** using SIGKILL.
- **I** – Kills all processes using SIGKILL (including **init**). *The system is unusable after issuing this System Request Key code.*
- **h** – Displays help text.

This feature is most beneficial when using a development kernel or when experiencing system freezes.



WARNING

The System Request Key feature is considered a security risk because an unattended console provides an attacker with access to the system. For this reason, it is turned off by default.

See **/usr/share/doc/kernel-doc-*kernel_version*/Documentation/sysrq.txt** for more information about the System Request Key.

- **tainted** – Indicates whether a non-GPL module is loaded.
 - **0** – No non-GPL modules are loaded.
 - **1** – At least one module without a GPL license (including modules with no license) is loaded.
 - **2** – At least one module was force-loaded with the command **insmod -f**.
- **threads-max** – Sets the maximum number of threads to be used by the kernel, with a default value of **2048**.
- **version** – Displays the date and time the kernel was last compiled. The first field in this file, such as **#3**, relates to the number of times a kernel was built from the source base.

E.3.9.4. /proc/sys/net/

This directory contains subdirectories concerning various networking topics. Various configurations at the time of kernel compilation make different directories available here, such as **ethernet/**, **ipv4/**, **ipx/**, and **ipv6/**. By altering the files within these directories, system administrators are able to adjust the network configuration on a running system.

Given the wide variety of possible networking options available with Linux, only the most common `/proc/sys/net/` directories are discussed.

The `/proc/sys/net/core/` directory contains a variety of settings that control the interaction between the kernel and networking layers. The most important of these files are:

- **message_burst** – Sets the maximum number of new warning messages to be written to the kernel log in the time interval defined by **message_cost**. The default value of this file is **10**.

In combination with **message_cost**, this setting is used to enforce a rate limit on warning messages written to the kernel log from the networking code and mitigate *Denial of Service (DoS)* attacks. The idea of a DoS attack is to bombard the targeted system with requests that generate errors and either fill up disk partitions with log files or require all of the system's resources to handle the error logging.

The settings in **message_burst** and **message_cost** are designed to be modified based on the system's acceptable risk versus the need for comprehensive logging. For example, by setting **message_burst** to 10 and **message_cost** to 5, you allow the system to write the maximum number of 10 messages every 5 seconds.

- **message_cost** – Sets a cost on every warning message by defining a time interval for **message_burst**. The higher the value is, the more likely the warning message is ignored. The default value of this file is **5**.
- **netdev_max_backlog** – Sets the maximum number of packets allowed to queue when a particular interface receives packets faster than the kernel can process them. The default value for this file is **1000**.
- **optmem_max** – Configures the maximum ancillary buffer size allowed per socket.
- **rmem_default** – Sets the receive socket buffer default size in bytes.
- **rmem_max** – Sets the receive socket buffer maximum size in bytes.
- **wmem_default** – Sets the send socket buffer default size in bytes.
- **wmem_max** – Sets the send socket buffer maximum size in bytes.

The `/proc/sys/net/ipv4/` directory contains additional networking settings. Many of these settings, used in conjunction with one another, are useful in preventing attacks on the system or when using the system to act as a router.



WARNING

An erroneous change to these files may affect remote connectivity to the system.

The following is a list of some of the more important files within the `/proc/sys/net/ipv4/` directory:

- **icmp_echo_ignore_all** and **icmp_echo_ignore_broadcasts** – Allows the kernel to ignore ICMP ECHO packets from every host or only those originating from broadcast and multicast addresses, respectively. A value of **0** allows the kernel to respond, while a value of **1** ignores the packets.

- **ip_default_ttl** – Sets the default *Time To Live (TTL)*, which limits the number of hops a packet may make before reaching its destination. Increasing this value can diminish system performance.
- **ip_forward** – Permits interfaces on the system to forward packets. By default, this file is set to **0**. Setting this file to **1** enables network packet forwarding.
- **ip_local_port_range** – Specifies the range of ports to be used by TCP or UDP when a local port is needed. The first number is the lowest port to be used and the second number specifies the highest port. Any systems that expect to require more ports than the default 1024 to 4999 should use a range from 32768 to 61000.
- **tcp_syn_retries** – Provides a limit on the number of times the system re-transmits a SYN packet when attempting to make a connection.
- **tcp_retries1** – Sets the number of permitted re-transmissions attempting to answer an incoming connection. Default of **3**.
- **tcp_retries2** – Sets the number of permitted re-transmissions of TCP packets. Default of **15**.

The **/usr/share/doc/kernel-doc-*kernel_version*/Documentation/networking/ip-sysctl.txt** file contains a list of files and options available in the **/proc/sys/net/ipv4/** and **/proc/sys/net/ipv6/** directories. Use the **sysctl -a** command to list the parameters in the **sysctl** key format.

A number of other directories exist within the **/proc/sys/net/ipv4/** directory and each covers a different aspect of the network stack. The **/proc/sys/net/ipv4/conf/** directory allows each system interface to be configured in different ways, including the use of default settings for unconfigured devices (in the **/proc/sys/net/ipv4/conf/default/** subdirectory) and settings that override all special configurations (in the **/proc/sys/net/ipv4/conf/all/** subdirectory).



IMPORTANT

Red Hat Enterprise Linux 6 defaults to **strict** reverse path forwarding. Before changing the setting in the **rp_filter** file, see the entry on Reverse Path Forwarding in the [Red Hat Enterprise Linux 6 Security Guide](#) and [The Red Hat Knowledgebase article about rp_filter](#).

The **/proc/sys/net/ipv4/neigh/** directory contains settings for communicating with a host directly connected to the system (called a network neighbor) and also contains different settings for systems more than one hop away.

Routing over IPV4 also has its own directory, **/proc/sys/net/ipv4/route/**. Unlike **conf/** and **neigh/**, the **/proc/sys/net/ipv4/route/** directory contains specifications that apply to routing with any interfaces on the system. Many of these settings, such as **max_size**, **max_delay**, and **min_delay**, relate to controlling the size of the routing cache. To clear the routing cache, write any value to the **flush** file.

Additional information about these directories and the possible values for their configuration files can be found in:

/usr/share/doc/kernel-doc-*kernel_version*/Documentation/filesystems/proc.txt

E.3.9.5. **/proc/sys/vm/**

This directory facilitates the configuration of the Linux kernel's virtual memory (VM) subsystem. The kernel makes extensive and intelligent use of virtual memory, which is commonly referred to as swap space.

The following files are commonly found in the `/proc/sys/vm/` directory:

- **block_dump** – Configures block I/O debugging when enabled. All read/write and block dirtying operations done to files are logged accordingly. This can be useful if diagnosing disk spin up and spin downs for laptop battery conservation. All output when **block_dump** is enabled can be retrieved via **dmesg**. The default value is **0**.



NOTE

If **block_dump** is enabled at the same time as kernel debugging, it is prudent to stop the **klogd** daemon, as it generates erroneous disk activity caused by **block_dump**.

- **dirty_background_ratio** – Starts background writeback of dirty data at this percentage of total memory, via a pdflush daemon. The default value is **10**.
- **dirty_expire_centisecs** – Defines when dirty in-memory data is old enough to be eligible for writeout. Data which has been dirty in-memory for longer than this interval is written out next time a pdflush daemon wakes up. The default value is **3000**, expressed in hundredths of a second.
- **dirty_ratio** – Starts active writeback of dirty data at this percentage of total memory for the generator of dirty data, via pdflush. The default value is **20**.
- **dirty_writeback_centisecs** – Defines the interval between pdflush daemon wakeups, which periodically writes dirty in-memory data out to disk. The default value is **500**, expressed in hundredths of a second.
- **laptop_mode** – Minimizes the number of times that a hard disk needs to spin up by keeping the disk spun down for as long as possible, therefore conserving battery power on laptops. This increases efficiency by combining all future I/O processes together, reducing the frequency of spin ups. The default value is **0**, but is automatically enabled in case a battery on a laptop is used.

This value is controlled automatically by the acpid daemon once a user is notified battery power is enabled. No user modifications or interactions are necessary if the laptop supports the ACPI (Advanced Configuration and Power Interface) specification.

For more information, see the following installed documentation:

`/usr/share/doc/kernel-doc-kernel_version/Documentation/laptop-mode.txt`

- **max_map_count** – Configures the maximum number of memory map areas a process may have. In most cases, the default value of **65536** is appropriate.
- **min_free_kbytes** – Forces the Linux VM (virtual memory manager) to keep a minimum number of kilobytes free. The VM uses this number to compute a **pages_min** value for each **lowmem** zone in the system. The default value is in respect to the total memory on the machine.
- **nr_hugepages** – Indicates the current number of configured **hugetlb** pages in the kernel.

For more information, see the following installed documentation:

`/usr/share/doc/kernel-doc-kernel_version/Documentation/vm/hugetlbpage.txt`

- **nr_pflush_threads** – Indicates the number of pdflush daemons that are currently running. This file is read-only, and should not be changed by the user. Under heavy I/O loads, the default value of two is increased by the kernel.
- **overcommit_memory** – Configures the conditions under which a large memory request is accepted or denied. The following three modes are available:
 - **0** – The kernel performs heuristic memory over commit handling by estimating the amount of memory available and failing requests that are blatantly invalid. Unfortunately, since memory is allocated using a heuristic rather than a precise algorithm, this setting can sometimes allow available memory on the system to be overloaded. This is the default setting.
 - **1** – The kernel performs no memory over commit handling. Under this setting, the potential for memory overload is increased, but so is performance for memory intensive tasks (such as those executed by some scientific software).
 - **2** – The kernel fails any request for memory that would cause the total address space to exceed the sum of the allocated swap space and the percentage of physical RAM specified in **/proc/sys/vm/overcommit_ratio**. This setting is best for those who desire less risk of memory overcommitment.



NOTE

This setting is only recommended for systems with swap areas larger than physical memory.

- **overcommit_ratio** – Specifies the percentage of physical RAM considered when **/proc/sys/vm/overcommit_memory** is set to **2**. The default value is **50**.
- **page-cluster** – Sets the number of pages read in a single attempt. The default value of **3**, which actually relates to 16 pages, is appropriate for most systems.
- **swappiness** – Determines how much a machine should swap. The higher the value, the more swapping occurs. The default value, as a percentage, is set to **60**.

All kernel-based documentation can be found in the following locally installed location:

/usr/share/doc/kernel-doc-*kernel_version*/Documentation/, which contains additional information.

E.3.10. **/proc/sysvipc/**

This directory contains information about System V IPC resources. The files in this directory relate to System V IPC calls for messages (**msg**), semaphores (**sem**), and shared memory (**shm**).

E.3.11. **/proc/tty**

This directory contains information about the available and currently used *tty devices* on the system. Originally called *teletype devices*, any character-based data terminals are called *tty devices*.

In Linux, there are three different kinds of *tty devices*. *Serial devices* are used with serial connections, such as over a modem or using a serial cable. *Virtual terminals* create the common console connection, such as the virtual consoles available when pressing **Alt+<F-key>** at the system console. *Pseudo terminals* create a two-way communication that is used by some higher level applications, such as XFree86. The **drivers** file is a list of the current *tty devices* in use, as in the following example:

```

serial      /dev/cua    5 64-127 serial:callout
serial      /dev/ttys    4 64-127 serial
pty_slave   /dev/pts    136 0-255 pty:slave
pty_master  /dev/ptm    128 0-255 pty:master
pty_slave   /dev/ttyp    3 0-255 pty:slave
pty_master  /dev/pty    2 0-255 pty:master
/dev/vc/0    /dev/vc/0   4 0 system:vtmaster
/dev/ptmx   /dev/ptmx   5 2 system
/dev/console /dev/console 5 1 system:console
/dev/tty     /dev/tty    5 0 system:/dev/tty
unknown     /dev/vc/%d   4 1-63 console

```

The **/proc/tty/driver/serial** file lists the usage statistics and status of each of the serial tty lines.

In order for tty devices to be used as network devices, the Linux kernel enforces *line discipline* on the device. This allows the driver to place a specific type of header with every block of data transmitted over the device, making it possible for the remote end of the connection to treat a block of data as just one in a stream of data blocks. SLIP and PPP are common line disciplines, and each are commonly used to connect systems to one another over a serial link.

E.3.12. /proc/PID/

Out of Memory (OOM) refers to a computing state where all available memory, including swap space, has been allocated. When this situation occurs, it will cause the system to panic and stop functioning as expected. There is a switch that controls OOM behavior in **/proc/sys/vm/panic_on_oom**. When set to **1** the kernel will panic on OOM. A setting of **0** instructs the kernel to call a function named **oom_killer** on an OOM. Usually, **oom_killer** can kill rogue processes and the system will survive.

The easiest way to change this is to echo the new value to **/proc/sys/vm/panic_on_oom**.

```

# cat /proc/sys/vm/panic_on_oom
1

# echo 0 > /proc/sys/vm/panic_on_oom

# cat /proc/sys/vm/panic_on_oom
0

```

It is also possible to prioritize which processes get killed by adjusting the **oom_killer** score. In **/proc/PID/** there are two tools labeled **oom_adj** and **oom_score**. Valid scores for **oom_adj** are in the range -16 to +15. To see the current **oom_killer** score, view the **oom_score** for the process. **oom_killer** will kill processes with the highest scores first.

This example adjusts the **oom_score** of a process with a *PID* of 12465 to make it less likely that **oom_killer** will kill it.

```

# cat /proc/12465/oom_score
79872

# echo -5 > /proc/12465/oom_adj

# cat /proc/12465/oom_score
78

```

There is also a special value of -17, which disables **oom_killer** for that process. In the example below, **oom_score** returns a value of 0, indicating that this process would not be killed.

```
# cat /proc/12465/oom_score
78

# echo -17 > /proc/12465/oom_adj

# cat /proc/12465/oom_score
0
```

A function called **badness()** is used to determine the actual score for each process. This is done by adding up 'points' for each examined process. The process scoring is done in the following way:

1. The basis of each process's score is its memory size.
2. The memory size of any of the process's children (not including a kernel thread) is also added to the score
3. The process's score is increased for 'niced' processes and decreased for long running processes.
4. Processes with the **CAP_SYS_ADMIN** and **CAP_SYS_RAWIO** capabilities have their scores reduced.
5. The final score is then bitshifted by the value saved in the **oom_adj** file.

Thus, a process with the highest **oom_score** value will most probably be a non-privileged, recently started process that, along with its children, uses a large amount of memory, has been 'niced', and handles no raw I/O.

E.4. USING THE SYSCTL COMMAND

The **/sbin/sysctl** command is used to view, set, and automate kernel settings in the **/proc/sys** directory.

For a quick overview of all settings configurable in the **/proc/sys** directory, type the **/sbin/sysctl -a** command as root. This creates a large, comprehensive list, a small portion of which looks something like the following:

```
net.ipv4.route.min_pmtu = 552
kernel.sysrq = 0
kernel.sem = 250    32000   32    128
```

This is the same information seen if each of the files were viewed individually. The only difference is the file location. For example, the **/proc/sys/net/ipv4/route/min_pmtu** file is listed as **net.ipv4.route.min_pmtu**, with the directory slashes replaced by dots and the **proc.sys** portion assumed.

The **sysctl** command can be used in place of **echo** to assign values to writable files in the **/proc/sys** directory. For example, instead of using the command

```
echo 1 > /proc/sys/kernel/sysrq
```

use the equivalent **sysctl** command as follows:

```
sysctl -w kernel.sysrq="1"
kernel.sysrq = 1
```

While quickly setting single values like this in **/proc/sys** is helpful during testing, this method does not work as well on a production system as special settings within **/proc/sys** are lost when the machine is rebooted. To preserve custom settings, add them to the **/etc/sysctl.conf** file.

The **/etc/sysctl.conf** file is installed by the **initscripts** package to override some kernel default values and therefore only contains a few of the possible parameters. Use the **sysctl -a** command to list the parameters in the **sysctl** key format. See the **/usr/share/doc/kernel-doc-kernel_version/Documentation/networking/ip-sysctl.txt** file for more information on the possible settings.

Each time the system boots, the **init** program runs the **/etc/rc.d/rc.sysinit** script. This script contains a command to execute **sysctl** using **/etc/sysctl.conf** to determine the values passed to the kernel. Any values added to **/etc/sysctl.conf** therefore take effect each time the system boots. Note that modules loaded after **sysctl** has parsed this file might override the settings.

E.5. ADDITIONAL RESOURCES

Below are additional sources of information about the **proc** file system.

Installable Documentation

- **/usr/share/doc/kernel-doc-kernel_version/Documentation/** – This directory, which is provided by the **kernel-doc** package, contains documentation about the **proc** file system. Before accessing the kernel documentation, you must run the following command as root:

```
~]# yum install kernel-doc
```

- **/usr/share/doc/kernel-doc-kernel_version/Documentation/filesystems/proc.txt** – Contains assorted, but limited, information about all aspects of the **/proc** directory.
- **/usr/share/doc/kernel-doc-kernel_version/Documentation/sysrq.txt** – An overview of System Request Key options.
- **/usr/share/doc/kernel-doc-kernel_version/Documentation/sysctl/** – A directory containing a variety of **sysctl** tips, including modifying values that concern the kernel (**kernel.txt**), accessing file systems (**fs.txt**), and virtual memory use (**vm.txt**).
- **/usr/share/doc/kernel-doc-kernel_version/Documentation/networking/ip-sysctl.txt** – A detailed overview of IP networking options.

APPENDIX F. REVISION HISTORY

Revision 9-3	Wed Mar 15 2017	Mirek Jahoda
Version for 6.9 GA publication.		
Revision 8-3	Mon May 30 2016	Maxim Svistunov
The /proc/meminfo appendix section updated; made minor improvements.		
Revision 8-2	Wed May 25 2016	Maxim Svistunov
Added <i>Relax-and-Recover (ReaR)</i> ; made minor improvements.		
Revision 8-1	Thu May 10 2016	Maxim Svistunov
Red Hat Enterprise Linux 6.8 GA release of the Deployment Guide.		
Revision 7-1	Tue Jul 14 2015	Barbora Ančincová
Red Hat Enterprise Linux 6.7 GA release of the Deployment Guide.		
Revision 7-0	Fri Apr 17 2015	Barbora Ančincová
Red Hat Enterprise Linux 6.7 Beta release of the Deployment Guide.		
Revision 6-3	Thu Apr 2 2015	Barbora Ančincová
Updated <i>TigerVNC</i> , <i>Viewing and Managing Log Files</i> , <i>Registering the System and Managing Subscriptions</i> , and <i>The kdump Crash Recovery Service</i> .		
Revision 6-2	Fri Oct 14 2014	Barbora Ančincová
Red Hat Enterprise Linux 6.6 GA release of the Deployment Guide.		
Revision 6-1	Fri Aug 22 2014	Jaromír Hradílek
Updated <i>NetworkManager</i> , <i>Network Interfaces</i> , <i>Configuring Authentication</i> , <i>The kdump Crash Recovery Service</i> , and <i>The proc File System</i> .		
Revision 6-0	Mon Aug 11 2014	Jaromír Hradílek
Red Hat Enterprise Linux 6.6 Beta release of the Deployment Guide.		
Revision 5-1	Thu Nov 21 2013	Jaromír Hradílek
Red Hat Enterprise Linux 6.5 GA release of the Deployment Guide.		
Revision 5-0	Thu Oct 3 2013	Jaromír Hradílek
Red Hat Enterprise Linux 6.5 Beta release of the Deployment Guide.		
Revision 4-1	Thu Feb 21 2013	Jaromír Hradílek
Red Hat Enterprise Linux 6.4 GA release of the Deployment Guide.		
Revision 4-0	Thu Dec 6 2012	Jaromír Hradílek
Red Hat Enterprise Linux 6.4 Beta release of the Deployment Guide.		
Revision 3-1	Wed Jun 20 2012	Jaromír Hradílek
Red Hat Enterprise Linux 6.3 GA release of the Deployment Guide.		
Revision 3-0	Tue Apr 24 2012	Jaromír Hradílek
Red Hat Enterprise Linux 6.3 Beta release of the Deployment Guide.		
Revision 2-1	Tue Dec 6 2011	Jaromír Hradílek
Red Hat Enterprise Linux 6.2 GA release of the Deployment Guide.		
Revision 2-0	Mon Oct 3 2011	Jaromír Hradílek

Red Hat Enterprise Linux 6.2 Beta release of the Deployment Guide.

Revision 1-1	Wed May 19 2011	Jaromír Hradílek
Red Hat Enterprise Linux 6.1 GA release of the Deployment Guide.		
Revision 1-0	Tue Mar 22 2011	Jaromír Hradílek
Red Hat Enterprise Linux 6.1 Beta release of the Deployment Guide.		
Revision 0-1	Tue Nov 9 2010	Douglas Silas
Red Hat Enterprise Linux 6.0 GA release of the Deployment Guide.		
Revision 0-0	Mon Nov 16 2009	Douglas Silas
Initialization of the Red Hat Enterprise Linux 6 Deployment Guide.		

INDEX

Symbols

.fetchmailrc, [Fetchmail Configuration Options](#)

server options, [Server Options](#)

user options, [User Options](#)

.htaccess , [Common httpd.conf Directives](#)

(see also Apache HTTP Server)

.htpasswd , [Common httpd.conf Directives](#)

(see also Apache HTTP Server)

.procmailrc, [Procmail Configuration](#)

/dev/oprofile/, [Understanding /dev/oprofile/](#)

/etc/named.conf (see BIND)

/etc/sysconfig/ directory (see sysconfig directory)

/etc/sysconfig/dhcpd, [Starting and Stopping the Server](#)

/proc/ directory (see proc file system)

/var/spool/anacron , [Configuring Anacron Jobs](#)

/var/spool/cron , [Configuring Cron Jobs](#)

(see OProfile)

A

Access Control

configuring in SSSD, [Creating Domains: Access Control](#)

SSSD rules, [Creating Domains: Access Control](#)

anacron, [Cron and Anacron](#)

anacron configuration file, [Configuring Anacron Jobs](#)

user-defined tasks, [Configuring Anacron Jobs](#)

anacrontab , [Configuring Anacron Jobs](#)

Apache HTTP Server

additional resources

installable documentation, [Additional Resources](#)

installed documentation, [Additional Resources](#)

useful websites, [Additional Resources](#)

checking configuration, [Editing the Configuration Files](#)

checking status, [Verifying the Service Status](#)

directives

<Directory> , [Common httpd.conf Directives](#)

<IfDefine> , [Common httpd.conf Directives](#)

<IfModule>, [Common httpd.conf Directives](#)
<Location>, [Common httpd.conf Directives](#)
<Proxy>, [Common httpd.conf Directives](#)
<VirtualHost>, [Common httpd.conf Directives](#)
AccessFileName, [Common httpd.conf Directives](#)
Action, [Common httpd.conf Directives](#)
AddDescription, [Common httpd.conf Directives](#)
AddEncoding, [Common httpd.conf Directives](#)
AddHandler, [Common httpd.conf Directives](#)
AddIcon, [Common httpd.conf Directives](#)
AddIconByEncoding, [Common httpd.conf Directives](#)
AddIconByType, [Common httpd.conf Directives](#)
AddLanguage, [Common httpd.conf Directives](#)
AddType, [Common httpd.conf Directives](#)
Alias, [Common httpd.conf Directives](#)
Allow, [Common httpd.conf Directives](#)
AllowOverride, [Common httpd.conf Directives](#)
BrowserMatch, [Common httpd.conf Directives](#)
CacheDefaultExpire, [Common httpd.conf Directives](#)
CacheDisable, [Common httpd.conf Directives](#)
CacheEnable, [Common httpd.conf Directives](#)
CacheLastModifiedFactor, [Common httpd.conf Directives](#)
CacheMaxExpire, [Common httpd.conf Directives](#)
CacheNegotiatedDocs, [Common httpd.conf Directives](#)
CacheRoot, [Common httpd.conf Directives](#)
CustomLog, [Common httpd.conf Directives](#)
DefaultIcon, [Common httpd.conf Directives](#)
DefaultType, [Common httpd.conf Directives](#)
Deny, [Common httpd.conf Directives](#)
DirectoryIndex, [Common httpd.conf Directives](#)
DocumentRoot, [Common httpd.conf Directives](#)
ErrorDocument, [Common httpd.conf Directives](#)
ErrorLog, [Common httpd.conf Directives](#)
ExtendedStatus, [Common httpd.conf Directives](#)
Group, [Common httpd.conf Directives](#)
HeaderName, [Common httpd.conf Directives](#)
HostnameLookups, [Common httpd.conf Directives](#)
Include, [Common httpd.conf Directives](#)
IndexIgnore, [Common httpd.conf Directives](#)
IndexOptions, [Common httpd.conf Directives](#)
KeepAlive, [Common httpd.conf Directives](#)

KeepAliveTimeout , [Common httpd.conf Directives](#)
LanguagePriority , [Common httpd.conf Directives](#)
Listen , [Common httpd.conf Directives](#)
LoadModule , [Common httpd.conf Directives](#)
LogFormat , [Common httpd.conf Directives](#)
LogLevel , [Common httpd.conf Directives](#)
MaxClients , [Common Multi-Processing Module Directives](#)
MaxKeepAliveRequests , [Common httpd.conf Directives](#)
MaxSpareServers , [Common Multi-Processing Module Directives](#)
MaxSpareThreads , [Common Multi-Processing Module Directives](#)
MinSpareServers , [Common Multi-Processing Module Directives](#)
MinSpareThreads , [Common Multi-Processing Module Directives](#)
NameVirtualHost , [Common httpd.conf Directives](#)
Options , [Common httpd.conf Directives](#)
Order , [Common httpd.conf Directives](#)
PidFile , [Common httpd.conf Directives](#)
ProxyRequests , [Common httpd.conf Directives](#)
ReadmeName , [Common httpd.conf Directives](#)
Redirect , [Common httpd.conf Directives](#)
ScriptAlias , [Common httpd.conf Directives](#)
ServerAdmin , [Common httpd.conf Directives](#)
ServerName , [Common httpd.conf Directives](#)
ServerRoot , [Common httpd.conf Directives](#)
ServerSignature , [Common httpd.conf Directives](#)
ServerTokens , [Common httpd.conf Directives](#)
SetEnvIf , [Common ssl.conf Directives](#)
StartServers , [Common Multi-Processing Module Directives](#)
SuexecUserGroup , [Common httpd.conf Directives](#)
ThreadsPerChild , [Common Multi-Processing Module Directives](#)
Timeout , [Common httpd.conf Directives](#)
TypesConfig , [Common httpd.conf Directives](#)
UseCanonicalName , [Common httpd.conf Directives](#)
User , [Common httpd.conf Directives](#)
UserDir , [Common httpd.conf Directives](#)

directories

/etc/httpd/ , [Common httpd.conf Directives](#)
/etc/httpd/conf.d/ , [Editing the Configuration Files](#), [Common httpd.conf Directives](#)
/usr/lib/httpd/modules/ , [Common httpd.conf Directives](#), [Working with Modules](#)
/usr/lib64/httpd/modules/ , [Common httpd.conf Directives](#), [Working with Modules](#)
/var/cache/mod_proxy/ , [Common httpd.conf Directives](#)
/var/www/cgi-bin/ , [Common httpd.conf Directives](#)

/var/www/html/ , [Common httpd.conf Directives](#)
/var/www/icons/ , [Common httpd.conf Directives](#)
~/public_html/ , [Common httpd.conf Directives](#)

files

.htaccess , [Common httpd.conf Directives](#)
.htpasswd , [Common httpd.conf Directives](#)
/etc/httpd/conf.d/nss.conf , [Enabling the mod_nss Module](#)
/etc/httpd/conf.d/ssl.conf , [Common ssl.conf Directives](#), [Enabling the mod_ssl Module](#)
/etc/httpd/conf/httpd.conf , [Editing the Configuration Files](#), [Common httpd.conf Directives](#),
[Common Multi-Processing Module Directives](#)
/etc/httpd/logs/access_log , [Common httpd.conf Directives](#)
/etc/httpd/logs/error_log , [Common httpd.conf Directives](#)
/etc/httpd/run/httpd.pid , [Common httpd.conf Directives](#)
/etc/mime.types , [Common httpd.conf Directives](#)

modules

developing, [Writing a Module](#)
loading, [Loading a Module](#)
mod_asis, [Notable Changes](#)
mod_cache, [New Features](#)
mod_cern_meta, [Notable Changes](#)
mod_disk_cache, [New Features](#)
mod_ext_filter, [Notable Changes](#)
mod_proxy_balancer, [New Features](#)
mod_rewrite , [Common httpd.conf Directives](#)
mod_ssl , [Setting Up an SSL Server](#)
mod_userdir, [Updating the Configuration](#)

restarting, [Restarting the Service](#)

SSL server

certificate, [An Overview of Certificates and Security](#), [Using an Existing Key and Certificate](#),
[Generating a New Key and Certificate](#)
certificate authority, [An Overview of Certificates and Security](#)
private key, [An Overview of Certificates and Security](#), [Using an Existing Key and Certificate](#),
[Generating a New Key and Certificate](#)
public key, [An Overview of Certificates and Security](#)

starting, [Starting the Service](#)

stopping, [Stopping the Service](#)

version 2.2

changes, [Notable Changes](#)
features, [New Features](#)
updating from version 2.0, [Updating the Configuration](#)

[virtual host](#), [Setting Up Virtual Hosts](#)

[at](#) , [At and Batch](#)

additional resources, [Additional Resources](#)

[authconfig](#) (see [Authentication Configuration Tool](#))

commands, [Configuring Authentication from the Command Line](#)

[authentication](#)

Authentication Configuration Tool, [Configuring System Authentication](#)

using fingerprint support, [Using Fingerprint Authentication](#)

using smart card authentication, [Enabling Smart Card Authentication](#)

[Authentication Configuration Tool](#)

and Kerberos authentication, [Using Kerberos with LDAP or NIS Authentication](#)

and LDAP, [Configuring LDAP Authentication](#)

and NIS, [Configuring NIS Authentication](#)

and Winbind, [Configuring Winbind Authentication](#)

and Winbind authentication, [Configuring Winbind Authentication](#)

[authoritative nameserver](#) (see [BIND](#))

[Automated Tasks](#), [Automating System Tasks](#)

B

[batch](#) , [At and Batch](#)

additional resources, [Additional Resources](#)

[Berkeley Internet Name Domain](#) (see [BIND](#))

[BIND](#)

additional resources

installed documentation, [Installed Documentation](#)

related books, [Related Books](#)

useful websites, [Useful Websites](#)

common mistakes, [Common Mistakes to Avoid](#)

configuration

acl statement, [Common Statement Types](#)

comment tags, [Comment Tags](#)

controls statement, [Other Statement Types](#)

include statement, [Common Statement Types](#)

key statement, [Other Statement Types](#)

logging statement, [Other Statement Types](#)

options statement, [Common Statement Types](#)

server statement, [Other Statement Types](#)

trusted-keys statement, [Other Statement Types](#)

view statement, [Other Statement Types](#)

zone statement, [Common Statement Types](#)

directories

[/etc/named/](#), [Configuring the named Service](#)

[/var/named/](#), [Editing Zone Files](#)

[/var/named/data/](#), [Editing Zone Files](#)

[/var/named/dynamic/](#), [Editing Zone Files](#)

[/var/named/slaves/](#), [Editing Zone Files](#)

features

Automatic Zone Transfer (AXFR), [Incremental Zone Transfers \(IXFR\)](#)

DNS Security Extensions (DNSSEC), [DNS Security Extensions \(DNSSEC\)](#)

Incremental Zone Transfer (IXFR), [Incremental Zone Transfers \(IXFR\)](#)

Internet Protocol version 6 (IPv6), [Internet Protocol version 6 \(IPv6\)](#)

multiple views, [Multiple Views](#)

Transaction SIGnature (TSIG), [Transaction SIGnatures \(TSIG\)](#)

files

[/etc/named.conf](#), [Configuring the named Service](#), [Configuring the Utility](#)

[/etc/rndc.conf](#), [Configuring the Utility](#)

[/etc/rndc.key](#), [Configuring the Utility](#)

resource record, [Nameserver Zones](#)

types

authoritative nameserver, [Nameserver Types](#)

primary (master) nameserver, [Nameserver Zones](#), [Nameserver Types](#)

recursive nameserver, [Nameserver Types](#)

secondary (slave) nameserver, [Nameserver Zones](#), [Nameserver Types](#)

utilities

dig, [BIND as a Nameserver](#), [Using the dig Utility](#), [DNS Security Extensions \(DNSSEC\)](#)

named, [BIND as a Nameserver](#), [Configuring the named Service](#)

rndc, [BIND as a Nameserver](#), [Using the rndc Utility](#)

zones

\$INCLUDE directive, [Common Directives](#)

\$ORIGIN directive, [Common Directives](#)

\$TTL directive, [Common Directives](#)

A (Address) resource record, [Common Resource Records](#)

CNAME (Canonical Name) resource record, [Common Resource Records](#)

comment tags, [Comment Tags](#)

description, [Nameserver Zones](#)

example usage, [A Simple Zone File](#), [A Reverse Name Resolution Zone File](#)

MX (Mail Exchange) resource record, [Common Resource Records](#)
NS (Nameserver) resource record, [Common Resource Records](#)
PTR (Pointer) resource record, [Common Resource Records](#)
SOA (Start of Authority) resource record, [Common Resource Records](#)

blkid, [Using the blkid Command](#)

block devices, [/proc/devices](#)

(see also [/proc/devices](#))

definition of, [/proc/devices](#)

bonding (see channel bonding)

boot loader

verifying, [Verifying the Boot Loader](#)

boot media, [Preparing to Upgrade](#)

C

ch-email .fetchmailrc

global options, [Global Options](#)

channel bonding

configuration, [Using Channel Bonding](#)

description, [Using Channel Bonding](#)

interface

configuration of, [Channel Bonding Interfaces](#)

parameters to bonded interfaces, [Bonding Module Directives](#)

channel bonding interface (see kernel module)

character devices, [/proc/devices](#)

(see also [/proc/devices](#))

definition of, [/proc/devices](#)

chkconfig (see services configuration)

Configuration File Changes, [Preserving Configuration File Changes](#)

CPU usage, [Viewing CPU Usage](#)

crash

analyzing the dump

message buffer, [Displaying the Message Buffer](#)

open files, [Displaying Open Files](#)

processes, [Displaying a Process Status](#)

stack trace, [Displaying a Backtrace](#)

virtual memory, [Displaying Virtual Memory Information](#)

opening the dump image, [Running the crash Utility](#)
system requirements, [Analyzing the Core Dump](#)

`createrepo`, [Creating a Yum Repository](#)
`cron`, [Cron and Anacron](#)
additional resources, [Additional Resources](#)
cron configuration file, [Configuring Cron Jobs](#)
user-defined tasks, [Configuring Cron Jobs](#)

`crontab` , [Configuring Cron Jobs](#)
CUPS (see [Printer Configuration](#))

D

`date` (see [date configuration](#))
`date configuration`
date, [Date and Time Setup](#)
`system-config-date`, [Date and Time Properties](#)

`default gateway`, [Static Routes and the Default Gateway](#)
deleting cache files
in SSSD, [Managing the SSSD Cache](#)

Denial of Service attack, [/proc/sys/net/](#)
(see also [/proc/sys/net/](#) directory)
definition of, [/proc/sys/net/](#)

desktop environments (see X)
`df`, [Using the df Command](#)
`DHCP`, [DHCP Servers](#)
additional resources, [Additional Resources](#)
client configuration, [Configuring a DHCPv4 Client](#)
command-line options, [Starting and Stopping the Server](#)
connecting to, [Configuring a DHCPv4 Client](#)
`dhcpd.conf`, [Configuration File](#)
`dhcpd.leases`, [Starting and Stopping the Server](#)
`dhcpd6.conf`, [DHCP for IPv6 \(DHCPv6\)](#)
DHCPv6, [DHCP for IPv6 \(DHCPv6\)](#)
`dhcrelay`, [DHCP Relay Agent](#)
global parameters, [Configuration File](#)
`group`, [Configuration File](#)
`options`, [Configuration File](#)
reasons for using, [Why Use DHCP?](#)
Relay Agent, [DHCP Relay Agent](#)
`shared-network`, [Configuration File](#)

starting the server, [Starting and Stopping the Server](#)
stopping the server, [Starting and Stopping the Server](#)
subnet, [Configuration File](#)

`dhcpd.conf`, [Configuration File](#)
`dhcpd.leases`, [Starting and Stopping the Server](#)
DHCPv4
server configuration, [Configuring a DHCPv4 Server](#)

`dhcrelay`, [DHCP Relay Agent](#)
`dig` (see BIND)
directory server (see OpenLDAP)
display managers (see X)
DNS
definition, [DNS Servers](#)
(see also BIND)

documentation
finding installed, [Practical and Common Examples of RPM Usage](#)

DoS attack (see Denial of Service attack)
downgrade
and SSSD, [Downgrading SSSD](#)

drivers (see kernel module)
DSA keys
generating, [Generating Key Pairs](#)

`du`, [Using the du Command](#)
Dynamic Host Configuration Protocol (see DHCP)

E

email
additional resources, [Additional Resources](#)
installed documentation, [Installed Documentation](#)
online documentation, [Online Documentation](#)
related books, [Related Books](#)

`Fetchmail`, [Fetchmail](#)

mail server

`Dovecot`, [Dovecot](#)

`Postfix`, [Postfix](#)

`Procmail`, [Mail Delivery Agents](#)

program classifications, [Email Program Classifications](#)

protocols, [Email Protocols](#)

IMAP, [IMAP](#)

POP, [POP](#)

SMTP, [SMTP](#)

security, [Securing Communication](#)

clients, [Secure Email Clients](#)

servers, [Securing Email Client Communications](#)

Sendmail, [Sendmail](#)

spam

filtering out, [Spam Filters](#)

types

Mail Delivery Agent, [Mail Delivery Agent](#)

Mail Transport Agent, [Mail Transport Agent](#)

Mail User Agent, [Mail User Agent](#)

epoch, [/proc/stat](#)

(see also [/proc/stat](#))

definition of, [/proc/stat](#)

Ethernet (see network)

Ethtool

command

devname, [Ethtool](#)

option

--advertise, [Ethtool](#)

--autoneg, [Ethtool](#)

--duplex, [Ethtool](#)

--features, [Ethtool](#)

--identify, [Ethtool](#)

--msglvl, [Ethtool](#)

--phyad, [Ethtool](#)

--port, [Ethtool](#)

--show-features, [Ethtool](#)

--show-time-stamping, [Ethtool](#)

--sopass, [Ethtool](#)

--speed, [Ethtool](#)

--statistics, [Ethtool](#)

--test, [Ethtool](#)

--wol, [Ethtool](#)
--xcvr, [Ethtool](#)

exec-shield

enabling, [/proc/sys/kernel/](#)
introducing, [/proc/sys/kernel/](#)

execution domains, [/proc/execdomains](#)

(see also [/proc/execdomains](#))
definition of, [/proc/execdomains](#)

extra packages for Enterprise Linux (EPEL)

installable packages, [Finding RPM Packages](#)

F

Fetchmail, [Fetchmail](#)

additional resources, [Additional Resources](#)
command options, [Fetchmail Command Options](#)
informational, [Informational or Debugging Options](#)
special, [Special Options](#)

configuration options, [Fetchmail Configuration Options](#)

global options, [Global Options](#)
server options, [Server Options](#)
user options, [User Options](#)

file system

virtual (see [proc file system](#))

file systems, [Viewing Block Devices and File Systems](#)

files, [proc file system](#)
changing, [Changing Virtual Files, Using the sysctl Command](#)
viewing, [Viewing Virtual Files, Using the sysctl Command](#)

findmnt, [Using the findmnt Command](#)

findsmb, [Connecting to a Samba Share](#)
findsmb program, [Samba Distribution Programs](#)

FQDN (see fully qualified domain name)

frame buffer device, [/proc/fb](#)
(see also [/proc/fb](#))

free, [Using the free Command](#)

FTP, [FTP](#)

(see also [vsftpd](#))

active mode, [The File Transfer Protocol](#)

command port, [The File Transfer Protocol](#)

data port, [The File Transfer Protocol](#)

definition of, [FTP](#)

introducing, [The File Transfer Protocol](#)

passive mode, [The File Transfer Protocol](#)

fully qualified domain name, [Nameserver Zones](#)

G

[gamin](#), [Monitoring Files and Directories with gamin](#)

[GNOME](#), [Desktop Environments](#)

(see also [X](#))

[gnome-system-log](#) (see [Log File Viewer](#))

[gnome-system-monitor](#), [Using the System Monitor Tool](#), [Using the System Monitor Tool](#), [Using the System Monitor Tool](#), [Using the System Monitor Tool](#)

[GnuPG](#)

checking RPM package signatures, [Checking a Package's Signature](#)

group configuration

modifying group properties, [Modifying Group Properties](#)

groups

additional resources, [Additional Resources](#)

installed documentation, [Installed Documentation](#)

GRUB boot loader

configuration file, [Configuring the GRUB Boot Loader](#)

configuring, [Configuring the GRUB Boot Loader](#)

H

hardware

viewing, [Viewing Hardware Information](#)

HTTP server (see [Apache HTTP Server](#))

[httpd](#) (see [Apache HTTP Server](#))

hugepages

configuration of, [/proc/sys/vm/](#)

I

[ifdown](#), [Interface Control Scripts](#)

ifup, [Interface Control Scripts](#)

information

about your system, [System Monitoring Tools](#)

initial RAM disk image

verifying, [Verifying the Initial RAM Disk Image](#)

IBM eServer System i, [Verifying the Initial RAM Disk Image](#)

initial RPM repositories

installable packages, [Finding RPM Packages](#)

insmod, [Loading a Module](#)

(see also kernel module)

installing package groups

installing package groups with PackageKit, [Installing and Removing Package Groups](#)

installing the kernel, [Manually Upgrading the Kernel](#)

K

KDE, [Desktop Environments](#)

(see also X)

kdump

additional resources

manual pages, [Additional Resources](#)

websites, [Additional Resources](#)

analyzing the dump (see crash)

configuring the service

default action, [Using the Kernel Dump Configuration Utility Configuring kdump on the Command Line](#)

dump image compression, [Using the Kernel Dump Configuration Utility Configuring kdump on the Command Line](#)

filtering level, [Using the Kernel Dump Configuration Utility Configuring kdump on the Command Line](#)

initial RAM disk, [Using the Kernel Dump Configuration Utility Configuring kdump on the Command Line](#)

kernel image, [Using the Kernel Dump Configuration Utility Configuring kdump on the Command Line](#)

kernel options, [Using the Kernel Dump Configuration Utility Configuring kdump on the Command Line](#)

memory usage, [Configuring kdump at First Boot Using the Kernel Dump Configuration Utility, Configuring kdump on the Command Line](#)

supported targets, [Using the Kernel Dump Configuration Utility Configuring kdump on the Command Line](#)

target location, [Using the Kernel Dump Configuration Utility](#) [Configuring kdump on the Command Line](#)

enabling the service, [Configuring kdump at First Boot](#) [Using the Kernel Dump Configuration Utility](#), [Configuring kdump on the Command Line](#)

fadump, [Using fadump on IBM PowerPC hardware](#)

installing, [Installing the kdump Service](#)

running the service, [Configuring kdump on the Command Line](#)

sadump, [Using sadump on Fujitsu PRIMEQUEST systems](#)

system requirements, [Configuring the kdump Service](#)

testing the configuration, [Testing the Configuration](#)

kernel

downloading, [Downloading the Upgraded Kernel](#)

installing kernel packages, [Manually Upgrading the Kernel](#)

kernel packages, [Overview of Kernel Packages](#)

package, [Manually Upgrading the Kernel](#)

performing kernel upgrade, [Performing the Upgrade](#)

RPM package, [Manually Upgrading the Kernel](#)

upgrade kernel available, [Downloading the Upgraded Kernel](#)

Red Hat network, [Downloading the Upgraded Kernel](#)

Security Errata, [Downloading the Upgraded Kernel](#)

upgrading

preparing, [Preparing to Upgrade](#)

working boot media, [Preparing to Upgrade](#)

upgrading the kernel, [Manually Upgrading the Kernel](#)

Kernel Dump Configuration (see kdump)

kernel module

bonding module, [Using Channel Bonding](#)

description, [Using Channel Bonding](#)

parameters to bonded interfaces, [Bonding Module Directives](#)

definition, [Working with Kernel Modules](#)

directories

/etc/sysconfig/modules/, [Persistent Module Loading](#)

/lib/modules/<kernel_version>/kernel/drivers/, [Loading a Module](#)

files

/proc/modules, [Listing Currently-Loaded Modules](#)

listing

currently loaded modules, [Listing Currently-Loaded Modules](#)

module information, [Displaying Information About a Module](#)

loading

at the boot time, [Persistent Module Loading](#)

for the current session, [Loading a Module](#)

module parameters

bonding module parameters, [Bonding Module Directives](#)

supplying, [Setting Module Parameters](#)

unloading, [Unloading a Module](#)

utilities

`insmod`, [Loading a Module](#)

`lsmod`, [Listing Currently-Loaded Modules](#)

`modinfo`, [Displaying Information About a Module](#)

`modprobe`, [Loading a Module, Unloading a Module](#)

`rmmmod`, [Unloading a Module](#)

kernel package

`kernel`

for single,multicore and multiprocessor systems, [Overview of Kernel Packages](#)

`kernel-devel`

kernel headers and makefiles, [Overview of Kernel Packages](#)

`kernel-doc`

documentation files, [Overview of Kernel Packages](#)

`kernel-firmware`

firmware files, [Overview of Kernel Packages](#)

`kernel-headers`

C header files files, [Overview of Kernel Packages](#)

`perf`

firmware files, [Overview of Kernel Packages](#)

kernel upgrading

preparing, [Preparing to Upgrade](#)

keyboard configuration, [Keyboard Configuration](#)

Keyboard Indicator applet, [Adding the Keyboard Layout Indicator](#)

Keyboard Preferences utility, [Changing the Keyboard Layout](#)

layout, [Changing the Keyboard Layout](#)

typing break, [Setting Up a Typing Break](#)

Keyboard Indicator (see keyboard configuration)
Keyboard Preferences (see keyboard configuration)
[kwin, Window Managers](#)
(see also X)

L

LDAP (see OpenLDAP)
[Log File Viewer, Managing Log Files in a Graphical Environment](#)
filtering, [Viewing Log Files](#)
monitoring, [Monitoring Log Files](#)
refresh rate, [Viewing Log Files](#)
searching, [Viewing Log Files](#)

[log files, Viewing and Managing Log Files](#)
(see also Log File Viewer)
description, [Viewing and Managing Log Files](#)
locating, [Locating Log Files](#)
monitoring, [Monitoring Log Files](#)
rotating, [Locating Log Files](#)
rsyslogd daemon, [Viewing and Managing Log Files](#)
viewing, [Viewing Log Files](#)

[logrotate, Locating Log Files](#)
[lsblk, Using the lsblk Command](#)
[lscpu, Using the lscpu Command](#)
[lsmod, Listing Currently-Loaded Modules](#)
(see also kernel module)

[lspci, Using the lspci Command /proc/bus/pci](#)
[lspcmcia, Using the lspcmcia Command](#)
[lsusb, Using the lsusb Command](#)

M

Mail Delivery Agent (see email)
Mail Transport Agent (see email) (see MTA)
[Mail Transport Agent Switcher, Mail Transport Agent \(MTA\) Configuration](#)
[Mail User Agent, Mail Transport Agent \(MTA\) Configuration](#) (see email)
MDA (see Mail Delivery Agent)
memory usage, [Viewing Memory Usage](#)
[metacity, Window Managers](#)
(see also X)

[modinfo, Displaying Information About a Module](#)

(see also kernel module)

modprobe, [Loading a Module, Unloading a Module](#)

(see also kernel module)

module (see kernel module)

module parameters (see kernel module)

MTA (see Mail Transport Agent)

setting default, [Mail Transport Agent \(MTA\) Configuration](#)

switching with Mail Transport Agent Switcher, [Mail Transport Agent \(MTA\) Configuration](#)

MUA, [Mail Transport Agent \(MTA\) Configuration](#) (see Mail User Agent)

Multihomed DHCP

host configuration, [Host Configuration](#)

server configuration, [Configuring a Multihomed DHCP Server](#)

mwm, [Window Managers](#)

(see also X)

N

named (see BIND)

nameserver (see DNS)

net program, [Samba Distribution Programs](#)

network

additional resources, [Additional Resources](#)

bridge

bridging, [Network Bridge](#)

commands

/sbin/ifdown, [Interface Control Scripts](#)

/sbin/ifup, [Interface Control Scripts](#)

/sbin/service network, [Interface Control Scripts](#)

configuration, [Interface Configuration Files](#)

configuration files, [Network Configuration Files](#)

functions, [Network Function Files](#)

interface configuration files, [Interface Configuration Files](#)

interfaces

802.1Q, [Setting Up 802.1Q VLAN Tagging](#)

alias, [Alias and Clone Files](#)

channel bonding, [Channel Bonding Interfaces](#)

clone, [Alias and Clone Files](#)

dialup, [Dialup Interfaces](#)

Ethernet, [Ethernet Interfaces](#)

ethtool, [Ethtool](#)
VLAN, [Setting Up 802.1Q VLAN Tagging](#)

scripts, [Network Interfaces](#)

Network Time Protocol (see NTP)

NIC
binding into single channel, [Using Channel Bonding](#)

nmblookup program, [Samba Distribution Programs](#)

NSCD
and SSSD, [Using NSCD with SSSD](#)

NTP
configuring, [Network Time Protocol Properties](#), [Network Time Protocol Setup](#)
[ntpd](#), [Network Time Protocol Properties](#), [Network Time Protocol Setup](#)
[ntpdate](#), [Network Time Protocol Setup](#)

[ntpd](#) (see NTP)
[ntpdate](#) (see NTP)
[ntsysv](#) (see services configuration)

O

[opannotate](#) (see OProfile)
[opcontrol](#) (see OProfile)

OpenLDAP
checking status, [Checking the Service Status](#)
client applications, [Overview of Common LDAP Client Applications](#)
configuration
database, [Changing the Database-Specific Configuration](#)
global, [Changing the Global Configuration](#)
overview, [OpenLDAP Server Setup](#)

directives
[olcAllows](#), [Changing the Global Configuration](#)
[olcConnMaxPending](#), [Changing the Global Configuration](#)
[olcConnMaxPendingAuth](#), [Changing the Global Configuration](#)
[olcDisallow](#)s, [Changing the Global Configuration](#)
[olcIdleTimeout](#), [Changing the Global Configuration](#)
[olcLogFile](#), [Changing the Global Configuration](#)
[olcReadOnly](#), [Changing the Database-Specific Configuration](#)
[olcReferral](#), [Changing the Global Configuration](#)
[olcRootDN](#), [Changing the Database-Specific Configuration](#)
[olcRootPW](#), [Changing the Database-Specific Configuration](#)

[olcSuffix, Changing the Database-Specific Configuration](#)

[olcWriteTimeout, Changing the Global Configuration](#)

directories

[/etc/openldap/slapd.d/, Configuring an OpenLDAP Server](#)

[/etc/openldap/slapd.d/cn=config/cn=schema/, Extending Schema](#)

features, [OpenLDAP Features](#)

files

[/etc/openldap/ldap.conf, Configuring an OpenLDAP Server](#)

[/etc/openldap/slapd.d/cn=config.ldif, Changing the Global Configuration](#)

[/etc/openldap/slapd.d/cn=config/olcDatabase={2}bdb.ldif, Changing the Database-Specific Configuration](#)

installation, [Installing the OpenLDAP Suite](#)

migrating authentication information, [Migrating Old Authentication Information to LDAP Format](#)

packages, [Installing the OpenLDAP Suite](#)

restarting, [Restarting the Service](#)

running, [Starting the Service](#)

schema, [Extending Schema](#)

stopping, [Stopping the Service](#)

terminology

[attribute, LDAP Terminology](#)

[entry, LDAP Terminology](#)

[LDIF, LDAP Terminology](#)

utilities, [Overview of OpenLDAP Server Utilities](#), [Overview of OpenLDAP Client Utilities](#)

[OpenSSH, OpenSSH, Main Features](#)

(see also [SSH](#))

[additional resources, Additional Resources](#)

[client, OpenSSH Clients](#)

[scp, Using the scp Utility](#)

[sftp, Using the sftp Utility](#)

[ssh, Using the ssh Utility](#)

DSA keys

[generating, Generating Key Pairs](#)

RSA keys

[generating, Generating Key Pairs](#)

RSA Version 1 keys

[generating, Generating Key Pairs](#)

server, [Starting an OpenSSH Server](#)
 starting, [Starting an OpenSSH Server](#)
 stopping, [Starting an OpenSSH Server](#)

ssh-add, [Configuring ssh-agent](#)
ssh-agent, [Configuring ssh-agent](#)
ssh-keygen
 DSA, [Generating Key Pairs](#)
 RSA, [Generating Key Pairs](#)
 RSA Version 1, [Generating Key Pairs](#)

using key-based authentication, [Using Key-Based Authentication](#)

OpenSSL

additional resources, [Additional Resources](#)
SSL (see SSL)
TLS (see TLS)

ophelp, [Setting Events to Monitor](#)

opreport (see OProfile)

OProfile, [OProfile](#)

/dev/oprofile/, [Understanding /dev/oprofile/](#)
additional resources, [Additional Resources](#)
configuring, [Configuring OProfile](#)
separating profiles, [Separating Kernel and User-space Profiles](#)

events

sampling rate, [Sampling Rate](#)
setting, [Setting Events to Monitor](#)

Java, [OProfile Support for Java](#)

monitoring the kernel, [Specifying the Kernel](#)

opannotate, [Using opannotate](#)

opcontrol, [Configuring OProfile](#)

--no-vmlinux, [Specifying the Kernel](#)
--start, [Starting and Stopping OProfile](#)
--vmlinux=, [Specifying the Kernel](#)

ophelp, [Setting Events to Monitor](#)

opreport, [Using opreport, Getting more detailed output on the modules](#)
on a single executable, [Using opreport on a Single Executable](#)

oprofiled, [Starting and Stopping OProfile](#)
log file, [Starting and Stopping OProfile](#)

overview of tools, [Overview of Tools](#)
reading data, [Analyzing the Data](#)
saving data, [Saving Data](#)
starting, [Starting and Stopping OProfile](#)
[SystemTap, OProfile and SystemTap](#)
unit mask, [Unit Masks](#)

oprofiled (see OProfile)
[oprof_start, Graphical Interface](#)
OS/400 boot loader
 configuration file, [Configuring the OS/400 Boot Loader](#)
 configuring, [Configuring the OS/400 Boot Loader](#)

P

package
 kernel RPM, [Manually Upgrading the Kernel](#)

PackageKit, [PackageKit](#)
 adding and removing, [Using Add/Remove Software](#)
 architecture, [PackageKit Architecture](#)
 installing and removing package groups, [Installing and Removing Package Groups](#)
 installing packages, [PackageKit](#)
 managing packages, [PackageKit](#)
PolicyKit
 authentication, [Updating Packages with Software Update](#)

uninstalling packages, [PackageKit](#)
updating packages, [PackageKit](#)
viewing packages, [PackageKit](#)
viewing transaction log, [Viewing the Transaction Log](#)

packages
 adding and removing with PackageKit, [Using Add/Remove Software](#)
 dependencies, [Unresolved Dependency](#)
 determining file ownership with, [Practical and Common Examples of RPM Usage](#)
 displaying packages
 yum info, [Displaying Package Information](#)

 displaying packages with Yum
 yum info, [Displaying Package Information](#)

 extra packages for Enterprise Linux (EPEL), [Finding RPM Packages](#)
 filtering with PackageKit, [Finding Packages with Filters](#)
 Development, [Finding Packages with Filters](#)

Free, [Finding Packages with Filters](#)
Hide subpackages, [Finding Packages with Filters](#)
Installed, [Finding Packages with Filters](#)
No filter, [Finding Packages with Filters](#)
Only available, [Finding Packages with Filters](#)
Only development, [Finding Packages with Filters](#)
Only end user files, [Finding Packages with Filters](#)
Only graphical, [Finding Packages with Filters](#)
Only installed, [Finding Packages with Filters](#)
Only native packages, [Finding Packages with Filters](#)
Only newest packages, [Finding Packages with Filters](#)

filtering with PackageKit for packages, [Finding Packages with Filters](#)
finding deleted files from, [Practical and Common Examples of RPM Usage](#)
finding RPM packages, [Finding RPM Packages](#)
initial RPM repositories, [Finding RPM Packages](#)
installing a package group with Yum, [Installing Packages](#)
installing and removing package groups, [Installing and Removing Package Groups](#)
installing packages with PackageKit, [PackageKit](#), [Installing and Removing Packages \(and Dependencies\)](#)
dependencies, [Installing and Removing Packages \(and Dependencies\)](#)

installing RPM, [Installing and Upgrading](#)
installing with Yum, [Installing Packages](#)
iRed Hat Enterprise Linux installation media, [Finding RPM Packages](#)

kernel
for single,multicore and multiprocessor systems, [Overview of Kernel Packages](#)

kernel-devel
kernel headers and makefiles, [Overview of Kernel Packages](#)

kernel-doc
documentation files, [Overview of Kernel Packages](#)

kernel-firmware
firmware files, [Overview of Kernel Packages](#)

kernel-headers
C header files files, [Overview of Kernel Packages](#)

listing packages with Yum
Glob expressions, [Listing Packages](#)
yum grouplist, [Listing Packages](#)
yum list all, [Listing Packages](#)
yum list available, [Listing Packages](#)

yum list installed, [Listing Packages](#)

yum repolist, [Listing Packages](#)

yum search, [Listing Packages](#)

locating documentation for, [Practical and Common Examples of RPM Usage](#)

managing packages with PackageKit, [PackageKit](#)

obtaining list of files, [Practical and Common Examples of RPM Usage](#)

packages and package groups, [Packages and Package Groups](#)

perf

firmware files, [Overview of Kernel Packages](#)

querying uninstalled, [Practical and Common Examples of RPM Usage](#)

removing, [Uninstalling](#)

removing package groups with Yum, [Removing Packages](#)

removing packages with PackageKit, [Installing and Removing Packages \(and Dependencies\)](#)

RPM, [RPM](#)

already installed, [Package Already Installed](#)

configuration file changes, [Configuration File Changes](#)

conflict, [Conflicting Files](#)

failed dependencies, [Unresolved Dependency](#)

freshening, [Freshening](#)

pristine sources, [RPM Design Goals](#)

querying, [Querying](#)

removing, [Uninstalling](#)

source and binary packages, [RPM](#)

tips, [Practical and Common Examples of RPM Usage](#)

uninstalling, [Uninstalling](#)

verifying, [Verifying](#)

searching packages with Yum

yum search, [Searching Packages](#)

setting packages with PackageKit

checking interval, [Updating Packages with Software Update](#)

uninstalling packages with PackageKit, [PackageKit](#)

uninstalling packages with Yum, [Removing Packages](#)

yum remove package_name, [Removing Packages](#)

updating currently installed packages

available updates, [Updating Packages with Software Update](#)

updating packages with PackageKit, [PackageKit](#)

PolicyKit, [Updating Packages with Software Update](#)

Software Update, Updating Packages with Software Update

upgrading RPM, [Installing and Upgrading](#)

viewing packages with PackageKit, [PackageKit](#)

viewing transaction log, [Viewing the Transaction Log](#)

viewing Yum repositories with PackageKit, [Refreshing Software Sources \(Yum Repositories\)](#)

Yum instead of RPM, [RPM](#)

pdbeedit program, [Samba Distribution Programs](#)

PolicyKit, [Updating Packages with Software Update](#)

Postfix, [Postfix](#)

default installation, [The Default Postfix Installation](#)

postfix, [Mail Transport Agent \(MTA\) Configuration](#)

prefdm (see X)

primary nameserver (see BIND)

Printer Configuration

CUPS, [Printer Configuration](#)

IPP Printers, [Adding an IPP Printer](#)

LDP/LPR Printers, [Adding an LPD/LPR Host or Printer](#)

Local Printers, [Adding a Local Printer](#)

New Printer, [Starting Printer Setup](#)

Print Jobs, [Managing Print Jobs](#)

Samba Printers, [Adding a Samba \(SMB\) printer](#)

Settings, [The Settings Page](#)

Sharing Printers, [Sharing Printers](#)

printers (see Printer Configuration)

proc file system

/proc/buddyinfo, [/proc/buddyinfo](#)

/proc/bus/ directory, [/proc/bus/](#)

/proc/bus/pci

viewing using lspci, [/proc/bus/pci](#)

/proc/cmdline, [/proc/cmdline](#)

/proc/cpuinfo, [/proc/cpuinfo](#)

/proc/crypto, [/proc/crypto](#)

/proc/devices

block devices, [/proc/devices](#)

character devices, [/proc/devices](#)

/proc/dma, [/proc/dma](#)

/proc/driver/ directory, [/proc/driver/](#)

/proc/execdomains, [/proc/execdomains](#)

/proc/fb, [/proc/fb](#)
/proc/filesystems, [/proc/filesystems](#)
/proc/fs/ directory, [/proc/fs](#)
/proc/interrupts, [/proc/interrupts](#)
/proc/iomem, [/proc/iomem](#)
/proc/ioports, [/proc/ioports](#)
/proc/irq/ directory, [/proc/irq/](#)
/proc/kcore, [/proc/kcore](#)
/proc/kmsg, [/proc/kmsg](#)
/proc/loadavg, [/proc/loadavg](#)
/proc/locks, [/proc/locks](#)
/proc/mdstat, [/proc/mdstat](#)
/proc/meminfo, [/proc/meminfo](#)
/proc/misc, [/proc/misc](#)
/proc/modules, [/proc/modules](#)
/proc/mounts, [/proc/mounts](#)
/proc/mtrr, [/proc/mtrr](#)
/proc/net/ directory, [/proc/net/](#)
/proc/partitions, [/proc/partitions](#)
/proc/PID/ directory, [/proc/PID/](#)
/proc/scsi/ directory, [/proc/scsi/](#)
/proc/self/ directory, [/proc/self/](#)
/proc/slabinfo, [/proc/slabinfo](#)
/proc/stat, [/proc/stat](#)
/proc/swaps, [/proc/swaps](#)
[/proc/sys/ directory, /proc/sys/](#), Using the sysctl Command
(see also sysctl)
/proc/sys/dev/ directory, [/proc/sys/dev/](#)
/proc/sys/fs/ directory, [/proc/sys/fs/](#)
/proc/sys/kernel/ directory, [/proc/sys/kernel/](#)
/proc/sys/kernel/exec-shield, [/proc/sys/kernel/](#)
/proc/sys/kernel/sysrq (see system request key)
/proc/sys/net/ directory, [/proc/sys/net/](#)
/proc/sys/vm/ directory, [/proc/sys/vm/](#)

/proc/sysrq-trigger, [/proc/sysrq-trigger](#)
/proc/sysvipc/ directory, [/proc/sysvipc/](#)
/proc/tty/ directory, [/proc/tty/](#)
/proc/uptime, [/proc/uptime](#)
/proc/version, [/proc/version](#)
additional resources, [Additional Resources](#)
installed documentation, [Additional Resources](#)

changing files within, [Changing Virtual Files /proc/sys/](#), [Using the sysctl Command](#)
files within, top-level, [Top-level Files within the proc File System](#)
introduced, [The proc File System](#)
process directories, [Process Directories](#)
subdirectories within, [Directories within /proc/](#)
viewing files within, [Viewing Virtual Files](#)

processes, [Viewing System Processes](#)

Procmail, [Mail Delivery Agents](#)

additional resources, [Additional Resources](#)
configuration, [Procmail Configuration](#)
recipes, [Procmail Recipes](#)
delivering, [Delivering vs. Non-Delivering Recipes](#)
examples, [Recipe Examples](#)
flags, [Flags](#)
local lockfiles, [Specifying a Local Lockfile](#)
non-delivering, [Delivering vs. Non-Delivering Recipes](#)
SpamAssassin, [Spam Filters](#)
special actions, [Special Conditions and Actions](#)
special conditions, [Special Conditions and Actions](#)

ps, [Using the ps Command](#)

R

RAM, [Viewing Memory Usage](#)

rcp, [Using the scp Utility](#)

ReaR

basic usage, [Basic ReaR Usage](#)

recursive nameserver (see BIND)

Red Hat Support Tool

getting support on the command line, [Accessing Support Using the Red Hat Support Tool](#)

Red Hat Enterprise Linux installation media

installable packages, [Finding RPM Packages](#)

Red Hat Subscription Management

subscription, [Registering the System and Attaching Subscriptions](#)

removing package groups

removing package groups with PackageKit, [Installing and Removing Package Groups](#)

resource record (see BIND)

rmmod, [Unloading a Module](#)

(see also kernel module)

rndc (see BIND)

root nameserver (see BIND)

rpcclient program, [Samba Distribution Programs](#)

RPM, [RPM](#)

additional resources, [Additional Resources](#)

already installed, [Package Already Installed](#)

basic modes, [Using RPM](#)

checking package signatures, [Checking a Package's Signature](#)

configuration file changes, [Configuration File Changes](#)

conf.rpmsave, [Configuration File Changes](#)

conflicts, [Conflicting Files](#)

dependencies, [Unresolved Dependency](#)

design goals, [RPM Design Goals](#)

powerful querying, [RPM Design Goals](#)

system verification, [RPM Design Goals](#)

upgradability, [RPM Design Goals](#)

determining file ownership with, [Practical and Common Examples of RPM Usage](#)

documentation with, [Practical and Common Examples of RPM Usage](#)

failed dependencies, [Unresolved Dependency](#)

file conflicts

resolving, [Conflicting Files](#)

file name, [Installing and Upgrading](#)

finding deleted files with, [Practical and Common Examples of RPM Usage](#)

finding RPM packages, [Finding RPM Packages](#)

freshening, [Freshening](#)

GnuPG, [Checking a Package's Signature](#)

installing, [Installing and Upgrading](#)

md5sum, [Checking a Package's Signature](#)

querying, [Querying](#)

querying for file list, [Practical and Common Examples of RPM Usage](#)

querying uninstalled packages, [Practical and Common Examples of RPM Usage](#)

tips, [Practical and Common Examples of RPM Usage](#)

uninstalling, [Uninstalling](#)

upgrading, [Installing and Upgrading](#)

verifying, [Verifying](#)

website, [Useful Websites](#)

RPM Package Manager (see RPM)

RSA keys

[generating, Generating Key Pairs](#)

RSA Version 1 keys

[generating, Generating Key Pairs](#)

rsyslog, Viewing and Managing Log Files

[actions, Actions](#)

[configuration, Basic Configuration of Rsyslog](#)

[debugging, Debugging Rsyslog](#)

[filters, Filters](#)

[global directives, Global Directives](#)

[log rotation, Log Rotation](#)

[modules, Using Rsyslog Modules](#)

[new configuration format, Using the New Configuration Format](#)

[queues, Working with Queues in Rsyslog](#)

[rulesets, Rulesets](#)

[templates, Templates](#)

runlevel (see services configuration)

S

sadump

[additional resources](#)

[installed documents, Additional Resources](#)

Samba (see Samba)

[Abilities, Introduction to Samba](#)

[Account Information Databases, Samba Account Information Databases](#)

[Idapsam, Samba Account Information Databases](#)

[Idapsam_compat, Samba Account Information Databases](#)

[mysqlsam, Samba Account Information Databases](#)

[Plain Text, Samba Account Information Databases](#)

[smbpasswd, Samba Account Information Databases](#)

[tdbsam, Samba Account Information Databases](#)

[xmlsam, Samba Account Information Databases](#)

[Additional Resources, Additional Resources](#)

[installed documentation, Additional Resources](#)

[related books, Additional Resources](#)

[useful websites, Additional Resources](#)

Backward Compatible Database Back Ends, Samba Account Information Databases

[Browsing, Samba Network Browsing](#)

[configuration, Configuring a Samba Server, Command-Line Configuration](#)

default, [Configuring a Samba Server](#)

CUPS Printing Support, [Samba with CUPS Printing Support](#)

CUPS smb.conf, [Simple smb.conf Settings](#)

daemon

nmbd, [Samba Daemons and Related Services](#)

overview, [Samba Daemons and Related Services](#)

smbd, [Samba Daemons and Related Services](#)

winbindd, [Samba Daemons and Related Services](#)

encrypted passwords, [Encrypted Passwords](#)

findsmb, [Connecting to a Samba Share](#)

graphical configuration, [Graphical Configuration](#)

Introduction, [Introduction to Samba](#)

Network Browsing, [Samba Network Browsing](#)

Domain Browsing, [Domain Browsing](#)

WINS, [WINS \(Windows Internet Name Server\)](#))

New Database Back Ends, [Samba Account Information Databases](#)

Programs, [Samba Distribution Programs](#)

findsmb, [Samba Distribution Programs](#)

net, [Samba Distribution Programs](#)

nmblookup, [Samba Distribution Programs](#)

pdbedit, [Samba Distribution Programs](#)

rpcclient, [Samba Distribution Programs](#)

smbcacls, [Samba Distribution Programs](#)

smbclient, [Samba Distribution Programs](#)

smbcontrol, [Samba Distribution Programs](#)

smbpasswd, [Samba Distribution Programs](#)

smbspool, [Samba Distribution Programs](#)

smbstatus, [Samba Distribution Programs](#)

smbtar, [Samba Distribution Programs](#)

testparm, [Samba Distribution Programs](#)

wbinfo, [Samba Distribution Programs](#)

Reference, [Samba](#)

Samba Printers, [Adding a Samba \(SMB\) printer](#)

Security Modes, [Samba Security Modes, User-Level Security](#)

Active Directory Security Mode, [User-Level Security](#)

Domain Security Mode, [User-Level Security](#)

Share-Level Security, [Share-Level Security](#)

User Level Security, [User-Level Security](#)

[Server Types](#), [Samba Server Types and the smb.conf File](#)

server types

 Domain Controller, [Domain Controller](#)

 Domain Member, [Domain Member Server](#)

 Stand Alone, [Stand-alone Server](#)

service

 conditional restarting, [Starting and Stopping Samba](#)

 reloading, [Starting and Stopping Samba](#)

 restarting, [Starting and Stopping Samba](#)

 starting, [Starting and Stopping Samba](#)

 stopping, [Starting and Stopping Samba](#)

share

 connecting to via the command line, [Connecting to a Samba Share](#)

 connecting to with Nautilus, [Connecting to a Samba Share](#)

 mounting, [Mounting the Share](#)

smb.conf, [Samba Server Types and the smb.conf File](#)

 Active Directory Member Server example, [Domain Member Server](#)

 Anonymous Print Server example, [Stand-alone Server](#)

 Anonymous Read Only example, [Stand-alone Server](#)

 Anonymous Read/Write example, [Stand-alone Server](#)

 NT4-style Domain Member example, [Domain Member Server](#)

 PDC using Active Directory, [Domain Controller](#)

 PDC using tdbSAM, [Domain Controller](#)

 Secure File and Print Server example, [Stand-alone Server](#)

smbclient, [Connecting to a Samba Share](#)

WINS, [WINS \(Windows Internet Name Server\)](#)

scp (see OpenSSH)

secondary nameserver (see BIND)

security plug-in (see Security)

Security-Related Packages

 updating security-related packages, [Updating Packages](#)

Sendmail, [Sendmail](#)

 additional resources, [Additional Resources](#)

 aliases, [Masquerading](#)

 common configuration changes, [Common Sendmail Configuration Changes](#)

 default installation, [The Default Sendmail Installation](#)

 LDAP and, [Using Sendmail with LDAP](#)

 limitations, [Purpose and Limitations](#)

masquerading, [Masquerading](#)
purpose, [Purpose and Limitations](#)
spam, [Stopping Spam](#)
with UUCP, [Common Sendmail Configuration Changes](#)

sendmail, [Mail Transport Agent \(MTA\) Configuration](#)
service (see services configuration)
services configuration, [Services and Daemons](#)
 chkconfig, [Using the chkconfig Utility](#)
 ntsysv, [Using the ntsysv Utility](#)
 runlevel, [Configuring the Default Runlevel](#)
 service, [Running Services](#)
 system-config-services, [Using the Service Configuration Utility](#)

sftp (see OpenSSH)
slab pools (see /proc/slabinfo)
slapd (see OpenLDAP)
smbcacls program, [Samba Distribution Programs](#)
smbclient, [Connecting to a Samba Share](#)
smbclient program, [Samba Distribution Programs](#)
smbcontrol program, [Samba Distribution Programs](#)
smbpasswd program, [Samba Distribution Programs](#)
smbspool program, [Samba Distribution Programs](#)
smbstatus program, [Samba Distribution Programs](#)
smbtar program, [Samba Distribution Programs](#)

SpamAssassin
 using with Procmail, [Spam Filters](#)

ssh (see OpenSSH)
SSH protocol
 authentication, [Authentication](#)
 configuration files, [Configuration Files](#)
 system-wide configuration files, [Configuration Files](#)
 user-specific configuration files, [Configuration Files](#)

connection sequence, [Event Sequence of an SSH Connection](#)
features, [Main Features](#)
insecure protocols, [Requiring SSH for Remote Connections](#)
layers
 channels, [Channels](#)
 transport layer, [Transport Layer](#)

port forwarding, [Port Forwarding](#)
requiring for remote login, [Requiring SSH for Remote Connections](#)

security risks, [Why Use SSH?](#)

version 1, [Protocol Versions](#)

version 2, [Protocol Versions](#)

X11 forwarding, [X11 Forwarding](#)

`ssh-add`, [Configuring ssh-agent](#)

SSL, [Setting Up an SSL Server](#)

(see also Apache HTTP Server)

SSL server (see Apache HTTP Server)

SSSD

and NSCD, [Using NSCD with SSSD](#)

configuration file

creating, [Setting up the sssd.conf File](#)

location, [Using a Custom Configuration File](#)

sections, [Creating the sssd.conf File](#)

downgrading, [Downgrading SSSD](#)

identity provider

local, [Creating the sssd.conf File](#)

Kerberos authentication, [Creating Domains: Kerberos Authentication](#)

LDAP domain, [Creating Domains: LDAP](#)

supported LDAP directories, [Creating Domains: LDAP](#)

Microsoft Active Directory domain, [Creating Domains: Active Directory](#), [Configuring Domains: Active Directory as an LDAP Provider \(Alternative\)](#)

proxy domain, [Creating Domains: Proxy](#)

sudo rules

rules stored per host, [Configuring Services: sudo](#)

`startx`, [Runlevel 3 \(see X\)](#)

(see also X)

static route, [Static Routes and the Default Gateway](#)

`stunnel`, [Securing Email Client Communications](#)

subscriptions, [Registering the System and Managing Subscriptions](#)

sysconfig directory

`/etc/sysconfig/apm-scripts/` directory, [Directories in the /etc/sysconfig/ Directory](#)

`/etc/sysconfig/arpwatch`, [/etc/sysconfig/arpwatch](#)

`/etc/sysconfig/authconfig`, [/etc/sysconfig/authconfig](#)

`/etc/sysconfig/autofs`, [/etc/sysconfig/autofs](#)

`/etc/sysconfig/cfq/` directory, [Directories in the /etc/sysconfig/ Directory](#)

`/etc/sysconfig/clock`, [/etc/sysconfig/clock](#)

/etc/sysconfig/dhcpd, /etc/sysconfig/dhcpd
/etc/sysconfig/firstboot, /etc/sysconfig/firstboot
/etc/sysconfig/init, /etc/sysconfig/init
/etc/sysconfig/ip6tables-config, /etc/sysconfig/ip6tables-config
/etc/sysconfig/kernel, /etc/sysconfig/kernel
/etc/sysconfig/keyboard, /etc/sysconfig/keyboard
/etc/sysconfig/ldap, /etc/sysconfig/ldap
/etc/sysconfig/named, /etc/sysconfig/named
/etc/sysconfig/network, /etc/sysconfig/network
/etc/sysconfig/network-scripts/ directory, [Network Interfaces](#), [Directories in the /etc/sysconfig/ Directory](#)

(see also network)

/etc/sysconfig/networking/ directory, [Directories in the /etc/sysconfig/ Directory](#)

/etc/sysconfig/ntp, /etc/sysconfig/ntp

/etc/sysconfig/quagga, /etc/sysconfig/quagga

/etc/sysconfig/radvd, /etc/sysconfig/radvd

/etc/sysconfig/rhn/ directory, [Directories in the /etc/sysconfig/ Directory](#)

/etc/sysconfig/samba, /etc/sysconfig/samba

/etc/sysconfig/saslauthd, /etc/sysconfig/saslauthd

/etc/sysconfig/selinux, /etc/sysconfig/selinux

/etc/sysconfig/sendmail, /etc/sysconfig/sendmail

/etc/sysconfig/spamassassin, /etc/sysconfig/spamassassin

/etc/sysconfig/squid, /etc/sysconfig/squid

/etc/sysconfig/system-config-users, /etc/sysconfig/system-config-users

/etc/sysconfig/vncservers, /etc/sysconfig/vncservers

/etc/sysconfig/xinetd, /etc/sysconfig/xinetd

additional information about, [The sysconfig Directory](#)

additional resources, [Additional Resources](#)

installed documentation, [Installed Documentation](#)

directories in, [Directories in the /etc/sysconfig/ Directory](#)

files found in, [Files in the /etc/sysconfig/ Directory](#)

sysctl

configuring with /etc/sysctl.conf, [Using the sysctl Command](#)

controlling /proc/sys/, [Using the sysctl Command](#)

SysRq (see system request key)

system analysis

OProfile (see OProfile)

system information

cpu usage, [Viewing CPU Usage](#)

file systems, [Viewing Block Devices and File Systems](#)
gathering, [System Monitoring Tools](#)
hardware, [Viewing Hardware Information](#)
memory usage, [Viewing Memory Usage](#)
processes, [Viewing System Processes](#)
currently running, [Using the top Command](#)

System Monitor, [Using the System Monitor Tool](#), [Using the System Monitor Tool](#), [Using the System Monitor Tool](#), [Using the System Monitor Tool](#)

system request key

enabling, [/proc/sys/](#)

System Request Key

definition of, [/proc/sys/](#)

setting timing for, [/proc/sys/kernel/](#)

system-config-authentication (see Authentication Configuration Tool)

system-config-date (see time configuration, date configuration)

system-config-kdump (see kdump)

system-config-services (see services configuration)

systems

registration, [Registering the System and Managing Subscriptions](#)

subscription management, [Registering the System and Managing Subscriptions](#)

T

testparm program, [Samba Distribution Programs](#)

time configuration

date, [Date and Time Setup](#)

synchronize with NTP server, [Network Time Protocol Properties](#), [Network Time Protocol Setup](#)

system-config-date, [Date and Time Properties](#)

time zone configuration, [Time Zone Properties](#)

TLB cache (see hugepages)

TLS, [Setting Up an SSL Server](#)

(see also Apache HTTP Server)

tool

Authentication Configuration Tool, [Configuring System Authentication](#)

top, [Using the top Command](#)

twm, [Window Managers](#)

(see also X)

U

updating currently installed packages

available updates, [Updating Packages with Software Update](#)

updating packages with PackageKit

PolicyKit, [Updating Packages with Software Update](#)

users

additional resources, [Additional Resources](#)

installed documentation, [Installed Documentation](#)

V

virtual file system (see proc file system)

virtual files (see proc file system)

virtual host (see Apache HTTP Server)

vsftpd

additional resources, [Additional Resources](#)

installed documentation, [Installed Documentation](#)

online documentation, [Online Documentation](#)

condrestart, [Starting and Stopping vsftpd](#)

configuration file

[/etc/vsftpd/vsftpd.conf](#), [vsftpd Configuration Options](#)

access controls, [Log In Options and Access Controls](#)

anonymous user options, [Anonymous User Options](#)

daemon options, [Daemon Options](#)

directory options, [Directory Options](#)

file transfer options, [File Transfer Options](#)

format of, [vsftpd Configuration Options](#)

local-user options, [Local-User Options](#)

logging options, [Logging Options](#)

login options, [Log In Options and Access Controls](#)

network options, [Network Options](#)

security options, [Security Options](#)

encrypting, [Encrypting vsftpd Connections Using TLS](#)

multihome configuration, [Starting Multiple Copies of vsftpd](#)

restarting, [Starting and Stopping vsftpd](#)

RPM

files installed by, [Files Installed with vsftpd](#)

securing, [Encrypting vsftpd Connections Using TLS](#), [SELinux Policy for vsftpd](#)

SELinux, [SELinux Policy for vsftpd](#)
 starting, [Starting and Stopping vsftpd](#)
 starting multiple copies of, [Starting Multiple Copies of vsftpd](#)
 status, [Starting and Stopping vsftpd](#)
 stopping, [Starting and Stopping vsftpd](#)
 TLS, [Encrypting vsftpd Connections Using TLS](#)

W

wbinfo program, [Samba Distribution Programs](#)
 web server (see [Apache HTTP Server](#))
 window managers (see [X](#))

X**X**

[/etc/X11/xorg.conf](#)
 Boolean values for, [The Structure of the Configuration](#)
[Device](#), [The Device section](#)
[DRI](#), [The DRI section](#)
[Files section](#), [The Files section](#)
[InputDevice section](#), [The InputDevice section](#)
 introducing, [The xorg.conf.d Directory](#), [The xorg.conf File](#)
[Monitor](#), [The Monitor section](#)
[Screen](#), [The Screen section](#)
[Section tag](#), [The Structure of the Configuration](#)
[ServerFlags section](#), [The ServerFlags section](#)
[ServerLayout section](#), [The ServerLayout Section](#)
 structure of, [The Structure of the Configuration](#)

additional resources, [Additional Resources](#)
 installed documentation, [Installed Documentation](#)
 useful websites, [Useful Websites](#)

configuration directory
[/etc/X11/xorg.conf.d](#), [The xorg.conf.d Directory](#)

configuration files
[/etc/X11/ directory](#), [X Server Configuration Files](#)
[/etc/X11/xorg.conf](#), [The xorg.conf File](#)
 options within, [X Server Configuration Files](#)
 server options, [The xorg.conf.d Directory](#), [The xorg.conf File](#)

desktop environments
[GNOME](#), [Desktop Environments](#)

KDE, Desktop Environments

display managers

configuration of preferred, [Runlevel 5](#)

definition of, [Runlevel 5](#)

GNOME, [Runlevel 5](#)

KDE, [Runlevel 5](#)

prefdm script, [Runlevel 5](#)

xdm, [Runlevel 5](#)

fonts

Fontconfig, [Fonts](#)

Fontconfig, adding fonts to, [Adding Fonts to Fontconfig](#)

FreeType, [Fonts](#)

introducing, [Fonts](#)

Xft, [Fonts](#)

introducing, [The X Window System](#)

runlevels

3, [Runlevel 3](#)

5, [Runlevel 5](#)

runlevels and, [Runlevels and X](#)

window managers

kwin, [Window Managers](#)

metacity, [Window Managers](#)

mwm, [Window Managers](#)

twm, [Window Managers](#)

X clients, [The X Window System, Desktop Environments and Window Managers](#)

desktop environments, [Desktop Environments](#)

startx command, [Runlevel 3](#)

window managers, [Window Managers](#)

xinit command, [Runlevel 3](#)

X server, [The X Window System](#)

features of, [The X Server](#)

X Window System (see X)

X.500 (see OpenLDAP)

X.500 Lite (see OpenLDAP)

xinit (see X)

Xorg (see Xorg)

Y

Yum

configuring plug-ins, [Enabling, Configuring, and Disabling Yum Plug-ins](#)
configuring Yum and Yum repositories, [Configuring Yum and Yum Repositories](#)
disabling plug-ins, [Enabling, Configuring, and Disabling Yum Plug-ins](#)
displaying packages
 yum info, [Displaying Package Information](#)

displaying packages with Yum
 yum info, [Displaying Package Information](#)

enabling plug-ins, [Enabling, Configuring, and Disabling Yum Plug-ins](#)
installing a package group with Yum, [Installing Packages](#)
installing with Yum, [Installing Packages](#)
listing packages with Yum
 Glob expressions, [Listing Packages](#)
 yum grouplist, [Listing Packages](#)
 yum list, [Listing Packages](#)
 yum list all, [Listing Packages](#)
 yum list available, [Listing Packages](#)
 yum list installed, [Listing Packages](#)
 yum repolist, [Listing Packages](#)

packages and package groups, [Packages and Package Groups](#)

plug-ins
 kabi, [Plug-in Descriptions](#)
 presto, [Plug-in Descriptions](#)
 product-id, [Plug-in Descriptions](#)
 refresh-packagekit, [Plug-in Descriptions](#)
 rhnplugin, [Plug-in Descriptions](#)
 search-disabled-repos, [Plug-in Descriptions](#)
 security, [Plug-in Descriptions](#)
 subscription-manager, [Plug-in Descriptions](#)
 yum-downloadonly, [Plug-in Descriptions](#)

repository, [Adding, Enabling, and Disabling a Yum Repository](#), [Creating a Yum Repository](#)

searching packages with Yum
 yum search, [Searching Packages](#)

setting [main] options, [Setting \[main\] Options](#)
setting [repository] options, [Setting \[repository\] Options](#)
uninstalling package groups with Yum, [Removing Packages](#)
uninstalling packages with Yum, [Removing Packages](#)

`yum remove package_name`, [Removing Packages](#)

variables, [Using Yum Variables](#)

`yum cache`, [Working with Yum Cache](#)

`yum clean`, [Working with Yum Cache](#)

`Yum plug-ins`, [Yum Plug-ins](#)

`Yum repositories`

configuring Yum and Yum repositories, [Configuring Yum and Yum Repositories](#)

`yum update`, [Upgrading the System Off-line with ISO and Yum](#)

`Yum repositories`

viewing Yum repositories with PackageKit, [Refreshing Software Sources \(Yum Repositories\)](#)

`Yum Updates`

checking for updates, [Checking For Updates](#)

updating a single package, [Updating Packages](#)

updating all packages and dependencies, [Updating Packages](#)

updating packages, [Updating Packages](#)

updating packages automatically, [Updating Packages](#)

updating security-related packages, [Updating Packages](#)