

---

# Fedora Core 5

## Managing Software with yum



Stuart Ellis

Edited by Paul W. Fields

Copyright © 2006 Red Hat, Inc. and others.

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

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

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

For guidelines on the permitted uses of the Fedora trademarks, refer to [https://fedoraproject.org/wiki/Legal:Trademark\\_guidelines](https://fedoraproject.org/wiki/Legal:Trademark_guidelines).

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

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

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

All other trademarks are the property of their respective owners.

### Abstract

Documentation for the yum software management system.

1. Introduction .....	2
1.1. Purpose .....	2

1.2. Audience .....	2
1.3. Using This Document .....	3
1.4. Additional Resources .....	3
2. Software Management Concepts .....	4
2.1. About Packages .....	4
2.2. About Repositories .....	4
2.3. About Dependencies .....	5
2.4. Understanding Package Names .....	5
3. Software Management Tools in Fedora Core .....	6
4. Managing Software with <b>yum</b> .....	7
4.1. Installing New Software with <b>yum</b> .....	8
4.2. Updating Software with <b>yum</b> .....	9
4.3. Removing Software with <b>yum</b> .....	10
5. Searching for Packages with <b>yum</b> .....	10
5.1. Searching by Package Name and Attributes .....	10
5.2. Advanced Searches .....	11
5.3. Understanding Matches .....	11
6. Updating Your System with <b>yum</b> .....	12
6.1. Automatically Updating Your System .....	12
7. Configuring Access to Software Repositories .....	12
7.1. Adding a Repository as a Package Source .....	12
7.2. Manually Authorizing Package Sources .....	13
7.3. Understanding Repository Compatibility .....	13
7.4. Disabling or Removing Package Sources .....	14
8. Installing Software from an Isolated Package .....	14
9. Customizing <b>yum</b> .....	15
9.1. Editing the <b>yum</b> Configuration .....	15
9.2. Working with <b>yum</b> Plugins .....	16
10. Working with <b>yum</b> Caching .....	17
10.1. Enabling the Caches .....	17
10.2. Using <b>yum</b> in Cache-only Mode .....	18
10.3. Clearing the <b>yum</b> Caches .....	18
11. Using <b>yum</b> with a Proxy Server .....	19
11.1. Configuring Proxy Server Access .....	19
11.2. Configuring Proxy Server Access for a Single User .....	19
12. Acknowledgments .....	20
<b>Index</b> .....	<b>20</b>

## 1. Introduction

### 1.1. Purpose

This document presents basic concepts of software management on Fedora systems. It outlines the major functions of **yum**, the recommended software management tool for Fedora.

### 1.2. Audience

This document is intended for Fedora users of all levels of experience.

## 1.3. Using This Document

This document is a reference for using **yum**. You may wish to read some or all of the sections, depending upon your needs and level of experience. If you are a new user, read [Section 2, “Software Management Concepts”](#) before using **yum** for the first time. Experienced Linux users should start with [Section 4, “Managing Software with yum”](#).



### Previous Versions of Fedora Core

This document describes the configuration of **yum** on current versions of Fedora Core. You must perform the additional step noted in [Section 7.2, “Manually Authorizing Package Sources”](#) to enable **yum** on Fedora Core 3.

Most of the examples in this document use the package **tsclient**, which is included with Fedora Core. The **tsclient** package provides an application for remote desktop access. If you install it successfully you may start the application by choosing **Applications** → **Internet** → **Terminal Server Client**. To use the examples, substitute the name of the relevant package for **tsclient**. The example commands for Fedora package groups use the **MySQL Database** group.



### Avoid Logging in with the Root Account

You do not need to log in with the **root** account in order to manage your Fedora system. All of the commands shown in this tutorial that require **root** access will prompt you for the **root** password. The example terminal commands use **su -c** to provide this facility.

Fedora Core includes a **yum** configuration that is suitable for independent systems with Internet access. You may use **yum** and related software on such systems without any additional configuration.

If your system is part of a managed network, consult your network administrators for advice. You may need to configure **yum** to use a network proxy server. [Section 11, “Using yum with a Proxy Server”](#) explains how to configure **yum** to use a proxy server. Administrators may also suggest or require that **yum** clients use specific package repositories. Refer to [Section 7, “Configuring Access to Software Repositories”](#) for instructions on how to configure access to repositories.

To improve performance and enable disconnected operations, activate the **yum** caches on your system. Refer to [Section 10, “Working with yum Caching”](#) for more information on the caching option.

## 1.4. Additional Resources

The **yum** utility has features and options not discussed in this document. Read the **man** pages for **yum(8)** and **yum.conf(5)** to learn more, using the following commands:

```
man yum man yum.conf
```

Other useful **yum** resources on the Internet include:

Project Web site

<http://linux.duke.edu/projects/yum/>

Users mailing list

<https://lists.dulug.duke.edu/mailman/listinfo/yum/>

Development mailing list

<https://lists.dulug.duke.edu/pipermail/yum-devel/>



### Check Bugzilla First

If you encounter a persistent error with a specific operation, visit <http://bugzilla.redhat.com> and review the bug reports for the package or packages involved. An error in a package may cause all **yum** operations that rely on that package to fail. Please file bug reports for Fedora packages, including **yum**, on this Bugzilla web site.

## 2. Software Management Concepts

### 2.1. About Packages

Fedora software and documentation is supplied in the form of files called *RPM packages*. Each package is a compressed archive containing product information, program files, icons, documentation and management scripts. Management applications use these files to safely locate, install, update and remove software. For example, the Fedora installation process uses the packages supplied with Fedora Core to build or upgrade a system to your requirements.

Packages also include a digital signature to prove their source. Software management utilities verify this digital signature by using a GPG *public key*. The **yum** and **rpm** utilities share a common *keyring* that stores all of the public keys for approved package sources. The system administrator configures these approved package sources.



### All Fedora Packages are Open Source Software

All of the software provided by the Fedora Project is open source software. You may download and install Fedora packages on as many systems as desired.

### 2.2. About Repositories

A *repository* is a prepared directory or Web site that contains software packages and index files. Software management utilities such as **yum** automatically locate and obtain the correct RPM packages from these repositories. This method frees you from having to manually find and install new applications or updates. You may use a single command to update all system software, or search for new software by specifying criteria.

A network of servers provide several repositories for each version of Fedora Core. The package management utilities in Fedora Core are already configured to use three of these repositories:

#### Base

The packages that make up a Fedora Core release, as it is on disc

#### Updates

Updated versions of packages that are provided in Base

#### Extras

Packages for a large selection of additional software



### Fedora Development Repositories

Fedora Core also includes settings for several alternative repositories. These provide packages for various types of test system, and replace one or more of the standard repositories. Only enable support for one of the following repositories if you test or develop Fedora software: **fedora-devel** (Rawhide), **fedora-extras-devel**, and **updates-testing**.

Third-party software developers also provide repositories for their Fedora compatible packages. To learn how to configure your Fedora system to use third-party repositories, read [Section 7, “Configuring Access to Software Repositories”](#).

You may also use the *package groups* provided by the Fedora repositories to manage related packages as sets. Some third-party repositories add packages to these groups, or provide their packages as additional groups.



### Available Package Groups

To view a list of all of the available package groups for your Fedora system, run the command `su -c 'yum grouplist'`.

Use repositories to ensure that you always receive current versions of software. If several versions of the same package are available, your management utility automatically selects the latest version.



### Installing Software not from a Repository

Install software using manual methods only when you are confident there is no repository which can currently provide it. You may have to manage that software with manual methods, instead of with Fedora software management utilities.

The **yum** commands shown in this document use repositories as package sources. Refer to [Section 8, “Installing Software from an Isolated Package”](#) for details of using **yum** to install software from a package file.

## 2.3. About Dependencies

Some of the files installed on a Fedora distribution are *libraries* which may provide functions to multiple applications. When an application requires a specific library, the package which contains that library is a *dependency*. To properly install a package, Fedora must first satisfy its dependencies. The dependency information for a RPM package is stored within the RPM file.

The **yum** utility uses package dependency data to ensure that all of requirements for an application are met during installation. It automatically installs the packages for any dependencies not already present on your system. If a new application has requirements that conflict with existing software, **yum** aborts without making any changes to your system.

## 2.4. Understanding Package Names

Each package file has a long name that indicates several key pieces of information. For example, this is the full name of a **tsclient** package:

```
tscclient-0.132-6.i386.rpm
```

Management utilities commonly refer to packages with one of three formats:

- Package name: **tscclient**
- Package name with version and release numbers: **tscclient-0.132-6**
- Package name with hardware architecture: **tscclient.i386**

For clarity, **yum** lists packages in the format **name.architecture**. Repositories also commonly store packages in separate directories by architecture. In each case, the hardware architecture specified for the package is the *minimum* type of machine required to use the package.

i386

Suitable for any current Intel-compatible computer

noarch

Compatible with all computer architectures

ppc

Suitable for PowerPC systems, such as Apple Power Macintosh

x86\_64

Suitable for 64-bit Intel-compatible processors, such as Opterons

Some software may be optimized for particular types of Intel-compatible machine. Separate packages may be provided for **i386**, **i586**, **i686** and **x86\_64** computers. A machine with at least an Intel Pentium, VIA C3 or compatible CPU may use **i586** packages. Computers with an Intel Pentium Pro and above, or a current model of AMD chip, may use **i686** packages.

Use the short name of the package for **yum** commands. This causes **yum** to automatically select the most recent package in the repositories that matches the hardware architecture of your computer.

Specify a package with other name formats to override the default behavior and force **yum** to use the package that matches that version or architecture. Only override **yum** when you know that the default package selection has a bug or other fault that makes it unsuitable for installation.



### Package Names

You may use any of the following formats to specify a package in a **yum** operation: *name*, *name.architecture*, *name-version*, *name-version-release*, *name-version-release.architecture*, and *epoch:name-version-release.architecture*.

## 3. Software Management Tools in Fedora Core

The **yum** utility is a complete software management system. Fedora Core also includes two graphical applications for software management that use **yum**. The **pup** utility provides an interface for updating software, and the **pirut** application enables you to add or remove software.

Both graphical tools appear in the **Applications** desktop menu. To update your system with **pup**, select **Applications** → **System Tools** → **Software Updater**. To add or remove software with **pirut**, select **Applications** → **Add/Remove Software**.

The **rpm** command-line utility has many functions for working with individual RPM packages. You may use it to manually install and remove packages from your system. If you install software with the **rpm** utility, you must manually check and install any dependencies. For this reason, **pirut** and **yum** are the recommended methods for installing software.



### Current Package Versions

The **pirut** and **yum** utilities ensure that you have the most recent version of software packages. Other methods do not guarantee that the packages are current.

## 4. Managing Software with **yum**

Use the **yum** utility to modify the software on your system in four ways:

- To install new software from package repositories
- To install new software from an individual package file
- To update existing software on your system
- To remove unwanted software from your system



### Installing Software from a Package File

The **yum** commands shown in this section use repositories as package sources. Refer to *Section 8, “Installing Software from an Isolated Package”* for details of using **yum** to install software from an individual package file.

To use **yum**, specify a function and one or more packages or package groups. Each section below gives some examples.

For each operation, **yum** downloads the latest package information from the configured repositories. If your system uses a slow network connection **yum** may require several seconds to download the repository indexes and the header files for each package.

The **yum** utility searches these data files to determine the best set of actions to produce the required result, and displays the transaction for you to approve. The transaction may include the installation, update, or removal of additional packages, in order to resolve software dependencies.

This is an example of the transaction for installing **tsclient**:

```
=====
Package Arch Version Repository Size
=====
Installing:
tsclient i386 0.132-6 base 247 k
Installing for dependencies:
```

```
rdesktop i386 1.4.0-2 base 107 k
```

### Transaction Summary

```
=====
Install 2 Package(s)
Update 0 Package(s)
Remove 0 Package(s)
Total download size: 355 k
Is this ok [y/N]:
```

#### Example 1. Format of **yum** Transaction Reports

Review the list of changes, and then press **y** to accept and begin the process. If you press **N** or **Enter**, **yum** does not download or change any packages.



### Package Versions

The **yum** utility only displays and uses the newest version of each package, unless you specify an older version.

The **yum** utility also imports the repository public key if it is not already installed on the **rpm** keyring.

This is an example of the public key import:

```
warning: rpmts_HdrFromFdno: Header V3 DSA signature: NOKEY, key ID 4f2a6fd2
public key not available for tsclient-0.132-6.i386.rpm
Retrieving GPG key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-fedora
Importing GPG key 0x4F2A6FD2 "Fedora Project <fedora@redhat.com>"
Is this ok [y/N]:
```

#### Example 2. Format of **yum** Public Key Import

Check the public key, and then press **y** to import the key and authorize the key for use. If you press **N** or **Enter**, **yum** stops without installing any packages.

To ensure that downloaded packages are genuine, **yum** verifies the digital signature of each package against the public key of the provider. Once all of the packages required for the transaction are successfully downloaded and verified, **yum** applies them to your system.



### Transaction Log

Every completed transaction records the affected packages in the log file `/var/log/yum.log`. You may only read this file with **root** access.

## 4.1. Installing New Software with yum

To install the package **tsclient**, enter the command:

```
su -c 'yum install tsclient'
```



Enter the password for the root account when prompted.

To install the package group **MySQL Database**, enter the command:

```
su -c 'yum groupinstall "MySQL Database"'
```

Enter the password for the root account when prompted.



### New Services Require Activation

When you install a service, Fedora does not activate or start it. To configure a new service to run on bootup, choose **Desktop** → **System Settings** → **Server Settings** → **Services**, or use the **chkconfig** and **service** command-line utilities.

## 4.2. Updating Software with yum

To update the **tsc1ient** package to the latest version, type:

```
su -c 'yum update tsc1ient'
```

Enter the password for the root account when prompted.



### New Software Versions Require Reloading

If a piece of software is in use when you update it, the old version remains active until the application or service is restarted. Kernel updates take effect when you reboot the system.



### Kernel Packages

Kernel packages remain on the system after they have been superseded by newer versions. This enables you to boot your system with an older kernel if an error occurs with the current kernel. To minimize maintenance, **yum** automatically removes obsolete kernel packages from your system, retaining only the current kernel and the previous version.

To update all of the packages in the package group **MySQL Database**, enter the command:

```
su -c 'yum groupupdate "MySQL Database"'
```

Enter the password for the root account when prompted.



### Updating the Entire System

To update all of the packages on your Fedora system, use the commands described in *Section 6, "Updating Your System with yum"*.

### 4.3. Removing Software with yum

To remove software, **yum** examines your system for both the specified software, and any software which claims it as a dependency. The transaction to remove the software deletes both the software and the dependencies.

To remove the **tsclient** package from your system, use the command:

```
su -c 'yum remove tsclient'
```

Enter the password for the root account when prompted.

To remove all of the packages in the package group **MySQL Database**, enter the command:

```
su -c 'yum groupremove "MySQL Database"'
```

Enter the password for the root account when prompted.



#### Data and Configuration File Retention

The removal process leaves user data in place but may remove configuration files in some cases. If a package removal does not include the configuration file, and you reinstall the package later, it may reuse the old configuration file.

## 5. Searching for Packages with yum

Use the search features of **yum** to find software that is available from the configured repositories, or already installed on your system. Searches automatically include both installed and available packages.

The format of the results depends upon the option. If the query produces no information, there are no packages matching the criteria.

### 5.1. Searching by Package Name and Attributes

To search for a specific package by name, use the **list** function. To search for the package **tsclient**, use the command:

```
su -c 'yum list tsclient'
```

Enter the password for the root account when prompted.

To make your queries more precise, specify packages with a name that include other attributes, such as version or hardware architecture. To search for version 0.132 of the application, use the command:

```
su -c 'yum list tsclient-0.132'
```



## Valid Package Attributes

Refer to *Section 2.4, “Understanding Package Names”* for information on package name formats and the attributes that they include.

## 5.2. Advanced Searches

If you do not know the name of the package, use the **search** or **provides** options. Alternatively, use wild cards with any **yum** search option to broaden the search criteria.

The **search** option checks the names, descriptions, summaries and listed package maintainers of all of the available packages to find those that match. For example, to search for all packages that relate to Palm Pilots, type:

```
su -c 'yum search PalmPilot'
```

Enter the password for the root account when prompted.

The **provides** function checks both the files included in the packages and the functions that the software provides. This option requires **yum** to download and read much larger index files than with the **search** option.

To search for all packages that include files called **libneon**, type:

```
su -c 'yum provides libneon'
```

To search for all packages that either provide a MTA (Mail Transport Agent) service, or include files with **mta** in their name:

```
su -c 'yum provides MTA'
```

For each command, at the prompt enter the password for the root account.

Use the standard wild-card characters to run any search option with a partial word or name: **?** to represent any one character, and **\*** to mean zero or more characters. Always add the escape character (**\**) before wild-cards.

To **list** all packages with names that begin with **tsc**, type:

```
su -c 'yum list tsc\*'
```

## 5.3. Understanding Matches

Searches with **yum** show all of the packages that match your criteria. Packages must meet the terms of the search exactly to be considered matches, unless you use wild-cards.

For example, a search query for **shadowutils** or **shadow-util** would not produce the package **shadow-utils**. This package would match and be shown if the query was **shadow-util\?**, or **shadow\\***.

## 6. Updating Your System with yum

Use the **update** option to upgrade all of your Fedora system software to the latest version with one operation.

To perform a full system update, type this command:

```
su -c 'yum update'
```

At the prompt, enter the root password.

### 6.1. Automatically Updating Your System

The **yum** package supplied with Fedora Core includes scripts to perform full system updates every day. To activate automatic daily updates, enter this command:

```
su -c '/sbin/chkconfig --level 345 yum on; /sbin/service yum start'
```

At the prompt, enter the password for the root account.



#### How Daily Updates are Run

There is no separate **yum** service that runs on your system. The command given above enables the control script `/etc/rc.d/init.d/yum`. This control script activates the script `/etc/cron.daily/yum.cron`, which causes the **cron** service to automatically begin a system update at 4am each day.

## 7. Configuring Access to Software Repositories

Fedora systems automatically use the Fedora Project repositories. These include Fedora Extras, the default source of packages for software that is not included with Fedora Core.



### Fedora Extras Repositories for Previous Versions of Fedora Core

You must manually configure Fedora Core 3 systems to use Fedora Extras, using the instructions at <http://fedora.redhat.com/projects/extras/>. For additional packages for Fedora Core 1 and Fedora Core 2, refer to <http://www.fedora.us/>.

If the Fedora Project does not supply packages for a product, the manufacturer may provide or recommend a separate repository. Members of the community also maintain repositories to provide packages for Fedora systems. For example, <http://www.jpackage.org/> distributes popular Java software as packages.

### 7.1. Adding a Repository as a Package Source

To add an extra repository, place a definition file in the `/etc/yum.repos.d/` directory on your system. Package providers make the definition files for their repositories available on their web sites.



### Definition File Extension

The names of repository definition files end with **.repo**.

You must have root access to add a file to the definitions directory. To copy the definition file **example.repo**, type this command:

```
su -c 'cp example.repo /etc/yum.repos.d/'
```

At the prompt, enter the password for the root account.

The configuration file for each repository should include a **gpgkey** setting. This setting specifies the location of a public key that verifies the packages provided by that repository. This public key is automatically imported the first time that you install software from the repository. If the configuration file provided does not include this setting, refer to [Section 7.2, “Manually Authorizing Package Sources”](#).

## 7.2. Manually Authorizing Package Sources

To manually add a public key to your **rpm** keyring, use the **import** feature of the **rpm** utility. To import the file **GPG-PUB-KEY.asc**, type the following command:

```
su -c 'rpm --import GPG-PUB-KEY.asc'
```

At the prompt, enter the password for the root account.

You may also import public keys directly from a web site. For example, to import the file **GPG-PUB-KEY.asc** on the web site [www.therepository.com](http://www.therepository.com) use this command:

```
su -c 'rpm --import http://www.therepository.com/GPG-PUB-KEY.asc'
```

At the prompt, enter the root password.



### Importing the Fedora Key on Fedora Core 3

To add the Fedora public key to the **rpm** keyring on Fedora Core 3 systems, run the command **su -c 'rpm --import /usr/share/rhn/RPM-GPG-KEY-fedora'**.

## 7.3. Understanding Repository Compatibility

The Fedora Extras repository provides packages which are built to the same standards as Fedora Core packages. Third-party packages should be compatible with these Fedora Project packages, unless the provider specifically states otherwise.

Always read the web site of the repository for information on package compatibility before you add it as a package source. Separate repository providers may offer different and incompatible versions of

the same software. Third-party repositories may also provide alternative packages for software that is included in Fedora repositories.

Alternative packages may contain versions of the software that function differently from the version in the Fedora Project packages. Determine the benefits and potential incompatibilities before replacing Fedora Project packages with alternative versions.



### Incompatible Repositories

If you configure your system to use incompatible repositories **yum** operations may fail.

Packages built for one version of Fedora are usually not compatible with other versions of Fedora. The web site of the provider should specifically state which versions of Fedora they support.



### Old Versions of yum and Current Repositories

The data format for repository indexes changed with version 2.1 of **yum**. This was the version supplied with Fedora Core 3. Repository providers should specify the versions of Fedora Core that they support. To confirm that an unlabeled repository is compatible with current versions of **yum**, check that it has a sub-directory called **repodata/**.

## 7.4. Disabling or Removing Package Sources

Set **enable=0** in a definition file to prevent **yum** from using that repository. The **yum** utility ignores any definition file with this setting.

To completely remove access to a repository:

1. Delete the relevant file from **/etc/yum.repos.d/**.
2. Delete the cache directory from **/var/cache/yum/**.

## 8. Installing Software from an Isolated Package

Use repositories and the standard **yum** commands to locate and install new software, unless the software package is not available from any repository. In these cases, use the **localinstall** function to install the software from the package file.



### Public Key is Required

Ensure that the public key for the package source has been imported before you install a package without a repository. Refer to *Section 7.2, "Manually Authorizing Package Sources"*.

Enter this command to install the package **tsclient-0.132-4.i386.rpm**:

```
su -c 'yum localinstall tsclient-0.132-4.i386.rpm'
```

At the prompt, enter the root password.



### Previously Installed Software is Updated

If the package provides a later version of software that is already installed on your system, **yum** updates the installed software.

If the package requires software that is not installed on your system, **yum** attempts to meet the dependencies with packages from the configured repositories. You may need to manually download and install additional packages in order to satisfy all of the dependencies.



### Maintaining Manually Installed Software

If you install software that is not provided by a repository, **yum update** cannot automatically upgrade it as new versions become available. To ensure that you have the latest packages, subscribe to e-mail or RSS services that notify you when new versions are released.

## 9. Customizing yum

To change the behavior of **yum**, you may either edit the configuration files, or install *plugins*. Plugins enable developers to add new features to **yum**.

### 9.1. Editing the yum Configuration

The file `/etc/yum.conf` provides the main configuration for **yum**. Settings in a repository definition file override the main configuration for those operations that use the defined repository.

To edit `/etc/yum.conf`, run a text editor with root privileges. This command opens `/etc/yum.conf` with **gedit**, the default text editor for Fedora desktop systems:

```
su -c 'gedit /etc/yum.conf'
```

Enter the password for the root account when prompted.

The main configuration file provides the settings that apply to all **yum** operations. These include caching options, and proxy server settings. The directory `/etc/yum.repos.d/` holds definition files for each repository that **yum** uses. Plugins use the configuration files in the directory `/etc/yum/pluginconf.d/`.

The following sections in this document provide further information on configuring **yum**:

- [Section 7, “Configuring Access to Software Repositories”](#)
- [Section 9.2, “Working with yum Plugins”](#)
- [Section 10.1, “Enabling the Caches”](#)
- [Section 11, “Using yum with a Proxy Server”](#)



### Further Documentation

Refer to the `man` page for `yum.conf` for a complete list of the configuration options supported by `yum`.

## 9.2. Working with yum Plugins

Each `yum` plugin is a single file, written in the Python programming language. You may download plugins from the `yum` project Web site, or from third-party providers. The `yum` project maintains a list of plugins on the page <http://wiki.linux.duke.edu/YumPlugins>.



### Plugin File Extension

The names of `yum` plugin files end with `.py`, the standard extension for Python scripts.

To install a plugin, copy it to the directory `/usr/lib/yum-plugins/`. Create a configuration file for the plugin in the directory `/etc/yum/pluginconf.d/`. Save the configuration file with the same name as the plugin, but with the extension `.conf`.



### root Privileges Required

You must have `root` access to add files to the directories `/usr/lib/yum-plugins/` and `/etc/yum/pluginconf.d/`.

For example, to copy the plugin `exampleplugin.py`, enter the command:

```
su -c 'cp exampleplugin.py /usr/lib/yum-plugins/'
```

Enter the password for the `root` account when prompted.

You may then create a configuration file for the plugin with a text editor. This example uses `gedit`, the default text editor for Fedora desktop systems:

```
su -c 'gedit /etc/yum/pluginconf.d/exampleplugin.conf'
```

Enter the password for the `root` account when prompted.

Each plugin configuration file includes the `enabled` setting. Some plugins also require additional settings. To determine the correct settings, either refer to the documentation supplied with the plugin, or read the plugin file itself with any text editor.

```
[main] enabled=1 anotheroption=0
```

[Example 3. Example Plugin Configuration File](#)





### Plugin Installed by Default

Fedora Core includes the **installonlyn** plugin. This plugin modifies **yum** to remove excess kernel packages, so that no more than a set number of kernels exist on the system. By default, **installonlyn** retains the two most current kernels, and automatically removes older kernel packages.

To remove a plugin, delete both the original file and the automatically generated bytecode file from **/usr/lib/yum-plugins/**. The bytecode file uses the same name as the plugin, but has the extension **.pyc**. Remove the relevant configuration file in **/etc/yum/pluginconf.d/**.

This command removes the plugin **exampleplugin**:

```
su -c 'rm -f /etc/yum/pluginconf.d/exampleplugin.conf; rm -f /usr/lib/yum-plugins/
exampleplugin.py*'
```

Enter the password for the root account when prompted.

## 10. Working with yum Caching

By default, current versions of **yum** delete the data files and packages that they download, after these have been successfully used for an operation. This minimizes the amount of storage space that **yum** uses. You may enable caching, so that **yum** retains the files that it downloads in cache directories.

Caches provide three advantages:

- The performance of **yum** increases
- You may carry out **yum** operations without a network connection, by using only the caches
- You may copy packages from the caches and reuse them elsewhere

By default, **yum** stores temporary files under the directory **/var/cache/yum/**, with one subdirectory for each configured repository. The **packages/** directory within each repository directory holds the cached packages. For example, the directory **/var/cache/yum/development/packages/** holds packages downloaded from the **development** repository.



### Clearing the yum Caches

Cached files use disk space until removed. You may wish to periodically clear the **yum** caches to recover capacity. Refer to [Section 10.3, “Clearing the yum Caches”](#) for information on clearing the caches.

If you remove a package from the cache, you do not affect the copy of the software installed on your system.

### 10.1. Enabling the Caches

To configure **yum** to retain downloaded files rather than discarding them, set the **keepcache** option in **/etc/yum.conf** to **1**:

```
keepcache=1
```

Refer to [Section 9.1, “Editing the yum Configuration”](#) for more information on editing the **yum** configuration file.

Once you enable caching, every **yum** operation may download package data from the configured repositories. To ensure that the caches have a set of package data, carry out an operation after you enable caching. Use a **list** or **search** query to download package data without modifying your system.

### 10.2. Using yum in Cache-only Mode

To carry out a **yum** command without a network connection, add the **-C** option. This causes **yum** to proceed without checking any network repositories, and use only cached files. In this mode, **yum** may only install packages that have been downloaded and cached by a previous operation.

To search for the package **tsclient** without using a network connection, enter the command:

```
su -c 'yum -C list tsclient'
```

Enter the password for the root account when prompted.



#### Cache-only Mode Requires Cached Data

Cache-only mode requires package data to exist in the caches. If you enable caching, every **yum** operation may update the data files, unless cache-only mode is specified for the operation.

### 10.3. Clearing the yum Caches

If you configure it to do so, **yum** retains the packages and package data files that it downloads, so that they may be reused in future operations without being downloaded again. To purge the package data files, use this command:

```
su -c 'yum clean headers'
```

Run this command to remove all of the packages held in the caches:

```
su -c 'yum clean packages'
```

When using these commands, at the prompt, enter the password for the root account.

Purging cached files causes those files to be downloaded again the next time that they are required. This increases the amount of time required to complete the operation.

## 11. Using yum with a Proxy Server

By default, **yum** accesses network repositories with HTTP. All **yum** HTTP operations use HTTP/1.1, and are compatible with web proxy servers that support this standard. You may also access FTP repositories, and configure **yum** to use an FTP proxy server. The **squid** package provides a proxy service for both HTTP/1.1 and FTP connections.



### Modifying yum for Network Compatibility

Refer to the **man** page for **yum.conf** for information on HTTP settings that may be modified for compatibility with nonstandard web proxy servers. Alternatively, configure **yum** to use an FTP proxy server, and access repositories that support FTP. The Fedora repositories support both HTTP and FTP.

### 11.1. Configuring Proxy Server Access

To enable all **yum** operations to use a proxy server, specify the proxy server details in **/etc/yum.conf**. The **proxy** setting must specify the proxy server as a complete URL, including the TCP port number. If your proxy server requires a username and password, specify these by adding **proxy\_username** and **proxy\_password** settings.

The settings below enable **yum** to use the proxy server `mycache.mydomain.com`, connecting to port 3128, with the username `yum-user` and the password `qwerty`.

```
# The proxy server - proxy server:port number
proxy=http://mycache.mydomain.com:3128
# The account details for yum connections
proxy_username=yum-user
proxy_password=qwerty
```

Example 4. Configuration File Settings for Using A Proxy Server



### Global Settings

If you define a proxy server in **/etc/yum.conf**, all users connect to the proxy server with those details when using **yum**.

### 11.2. Configuring Proxy Server Access for a Single User

To enable proxy access for a specific user, add the lines in the example box below to the user's shell profile. For the default **bash** shell, the profile is the file `~/.bash_profile`. The settings below enable **yum** to use the proxy server `mycache.mydomain.com`, connecting to port 3128.

```
# The Web proxy server used by this account
http_proxy="http://mycache.mydomain.com:3128"
export http_proxy
```

Example 5. Profile Settings for Using a Proxy Server

If the proxy server requires a username and password, add these to the URL. To include the username `yum-user` and the password `qwerty`, add these settings:

```
# The Web proxy server, with the username and password for this account
http_proxy="http://yum-user:qwerty@mycache.mydomain.com:3128"
export http_proxy
```

Example 6. Profile Settings for a Secured Proxy Server



### The http\_proxy Environment Variable

The `http_proxy` environment variable is also used by `curl` and other utilities. Although `yum` itself may use `http_proxy` in either upper-case or lower-case, `curl` requires the name of the variable to be in lower-case.

## 12. Acknowledgments

Paul Fields edited this document. Timothy Murphy reviewed the beta release.

# Index

## A

- Add/Remove Software utility, 6
- automatic updating, 12

## C

- caching, 17
  - cleaning caches, 18
  - enabling, 17
- configuration files, 15

## D

- dependencies
  - defined, 5

## F

- Fedora Extras, 12

## I

- installing software (see software, installing)
  - from a package (see software, installing from a package)

## L

- log file, 8

## P

- package groups
  - defined, 5
- packages
  - caching, 17
  - defined, 4

- hardware compatibility, 5
- locating, 10
- naming, 5
- software compatibility, 13
- plugins
  - installing, 16
  - installonlyn, 17
  - removing, 17
- proxy server, 19
- public keys
  - adding, 13

## R

- removing software, 10
- repositories
  - adding to yum, 12
  - compatibility, 13
  - defined, 4
  - disabling in yum, 14
  - finding, 12
  - removing from yum, 14
- rpm utility, 6

## S

- searching
  - for packages, 10
  - for repositories, 12
- software
  - installing, 8
  - installing from a package, 14
  - removing, 10
  - updating, 9
- Software Updater utility, 6

## U

- updating
  - full system, 12
  - software packages, 9

## Y

- yum
  - cache-only mode, 18
  - cleaning caches, 18
  - documentation, 3
  - mailing lists, 3
  - man pages, 3
  - software management, 7
  - updating full system, 12
  - using with a proxy server, 19
  - Web sites, 3

