Fedora Core 3

Software Management with yum

Stuart Ellis

Abstract

1. Introduction

1.1. Purpose

This tutorial 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 tutorial is intended for Fedora; users of all experience levels.

1.3. Using This Document

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. If you are an experienced Linux user, start with *Section 4, “Updating Your System with **`yum`**”*.

If you have several Fedora; systems on a network, you may benefit from setting up your own software repositories to manage the process of installation and updates. Refer to *Section 9, "Managing **yum** Repositories"* for details of maintaining your own repositories.

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`**.

Avoid Logging in with the Root Account

You do not need to log in with the root account in order to manage your Fedora Core; system. Any commands in this tutorial which require root access will prompt you for the root password. The procedures use the command **`su -c`** to provide this facility.

1.4. Additional Resources

The **`yum`** utility has features and options that are 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
```

The official home page for **`yum`** on the World Wide Web is *http://linux.duke.edu/projects/yum/*. The official mailing list for **`yum`** users is at *https://lists.dulug.duke.edu/mailman/listinfo/yum/*. The archive for the **`yum`** development mailing list is at *https://lists.dulug.duke.edu/pipermail/yum-devel/*.

2. Software Management Concepts

2.1. About Packages

Fedora; software and documentation is supplied in the form of RPM *packages*. Each package is a compressed archive which contains 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 with a GPG *public key*. The **`yum`** and **`rpm`** utilities share a common *keyring* which stores all public keys for the package sources approved by the system administrator.

2.2. About Repositories

A *repository* is a directory which contains prepared files which refer to software packages. Software management utilities such as **`yum`** automatically locate and obtain the correct RPM packages for an application 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 of your system's software, or search for new software by specifying criteria. In each case, the management utility connects to the configured repositories and checks files in each one to find the correct packages.

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

For these reasons, you should only manually install software when you are confident that no repository can currently provide it. If a piece of installed software is not available from a repository, you cannot automatically find or install newer versions. You must keep that product updated manually.

The package management utilities in Fedora Core; are automatically configured to use the network of repository servers maintained by the Fedora Project;. These repositories contain the software included with Fedora Core;, and a large selection of additional software known as Fedora Extras\;. Third-party software developers also provide repositories for their Fedora; compatible packages.

Open Source Software

All of the software provided by the Fedora Project; is open source software. For more information about open source software, refer to *http://www.opensource.org/*.

2.3. About Dependencies

You must consider package *dependencies* when manually installing software. For RPM software, a dependency is a capability provided by one package on which other packages rely. Some programs rely on external shared *libraries* to run properly. If a library is provided by an external package, that package may be a dependency for numerous other packages.

Management tools like **`yum`** use the information on dependencies stored within packages to ensure that all of the requirements are met when you install an application. The **`yum`** utility installs all required packages which are 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 package supplied with Fedora Core;:

```
tsclient-0.132-4.i386.rpm
```

Use only the name of the package itself with **`yum`**, except when it is necessary to specify the exact version or type. To specify the exact version of the application, use **`name-version`**. The package listings provided by **`yum`** use the format **`name.architecture`**, to specify the type of computer for which the package is intended.

These properties are valid for the file shown above:

- Package name: **`tsclient`**
- Package name with version number: **`tsclient-0.132`**
- Package name with hardware architecture: **`tsclient.i386`**

The hardware architecture is the *minimum* type of machine required for that specific package. Packages with architecture **`i386`** run on any current Intel-compatible computer. Packages for

PowerPC machines, such as Apple Macs, are indicated with **`ppc`**. Packages with architecture **`noarch`** have no architecture requirement.

Some software can be optimized for particular types of Intel-compatible machine, and separate packages may be provided for **`i386`**, **`i586`**, **`i686`** and **`x86_64`**. A computer with at least an Intel Pentium, VIA C3 or compatible CPU is an **`i586`**. Computers with an Intel Pentium II or later, or a current model of AMD chip, are **`i686`** machines. 64-bit PCs use **`x86_64`** packages for full 64-bit support.

Other Naming Conventions are Supported

Refer to *Section 5.1, “Searching by Package Name and Attributes”* for more information on specifying packages by name or type.

3. Software Management Tools in Fedora Core;

The **`yum`** utility is a complete software management system. Fedora Core; also includes other several other applications that can supplement **`yum`**.

On your desktop is an **Alert Icon** that keeps you informed about package updates. Until your system is updated, the icon appears as a red circle with a flashing exclamation mark. The **Alert Icon** is integrated with **up2date**, which enables you to easily install updates for your system.

Both **up2date** and **`yum`** are configured to use official Fedora; repositories. If you add other repositories to **`yum`**, for consistency you should also configure **up2date** to use them.

Repository Configuration

In Fedora Core; 4 and beyond, **up2date** is configured to automatically use repositories configured for **`yum`**. If you configure any new repositories for **`yum`**, **up2date** will use them also.

Also included in Fedora Core; is **system-config-packages**. To run this utility, rom the **Main Menu**, select **System Settings** → **Add/Remove Applications**. Unlike **up2date** and **`yum`**, this utility installs software packages from your Fedora Core; installation discs, and does not use repositories. This application is used on systems that do not have a network connection.

The **`rpm`** command-line utility has many functions for working with individual RPM packages. The **`rpm`** command can also be used to manually install and remove packages from your system. Installing software with the **`rpm`** utility can be difficult for novices, and is not recommended.

Current Package Versions

Using **up2date** and **`yum`** ensures that you have the most recent version of the packages that are being installed. Other methods do not guarantee that the packages are current.

4. Updating Your System with yum

To update all of your Fedora; system's software in a single operation, select **Applications** → **System Tools** → **Terminal** and type:

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

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

Data files are downloaded from each of the repositories that **yum** is configured to use. These index and header files are searched for information about newer versions of packages. On a slow connection the download process may take several seconds.

A list of all of the available updates for your system is displayed. Press **y** to accept the updates. If you accept the updates the relevant packages are then downloaded and installed.

4.1. Automatically Updating Your System

If your system is permanently connected to the network then updates can be performed at any time. The **yum** package includes scripts that can automatically carry out full updates every day.

To activate automatic daily updating, type this line:

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

Enter the root password when prompted.

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**, so that the **cron** service will perform the system update as one of the tasks that are automatically run each day.

5. Searching for Software with yum

You may use **yum** to find software that is available from the defined repositories, or is already installed on your system. Searches automatically include both installed and available packages.

Searches are not Case-sensitive

The **search** and **list** options of **yum** are not case-sensitive. For example, a query for **palmpilot** will automatically find **PalmPilot** packages.

5.1. Searching by Package Name and Attributes

To search for a specific package by name, use the **list** function. For example, to search for the package **tsclient** the command would be:

```
yum list tsclient
```

To make your queries more specific, add other package attributes. For example, to search for version 0.132 of the application the command would be:

```
yum list tsclient-0.132
```

Package Attributes

You may use any of the following formats for specifying a package in a `yum` query: `name`, `name.architecture`, `name-version`, `name-version-release`, `name-version-release.architecture`, and `epoch:name-version-release.architecture`.

5.2. Advanced Searches

If you do not know the name of the package, use either the `search` or `provides` options. `Search` 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 PalmPilots, type:

```
yum search PalmPilot
```

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 other types of search.

To search for all packages that include files called `libneon` you type:

```
yum provides libneon
```

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

```
yum provides MTA
```

Wildcards and Regular Expressions

You may use the standard wildcard characters in search criteria: `?` to represent any one character, and `*` to mean any characters. Use Perl or Python regular expressions to carry out more complex queries.

5.3. Understanding Matches

When carrying out a search **`yum`** shows all of the packages that match your criteria. Packages must meet the terms of the search exactly to be considered matches, unless you have used wildcards or a regular expression.

For example, querying for **`shadowutils`** or **`shadow-util`** would not produce the package **`shadow-utils`**. This package would match and be shown if the query was either **`shadow-util?`** or **`Shadow*`**.

When several versions of the same package are available, only the newest is used.

6. Managing Software with yum

The **`yum`** utility has four basic management functions:

- **`install`** new software from the repositories.
- **`update`** existing software.
- **`remove`** unwanted software.
- **`localinstall`**, to install software from a individual package.

In each case you must specify the function and the criteria. Some simple examples are given in each section.

Search Criteria

See *Section 5, "Searching for Software with **`yum`**"* for details of search criteria. The management options of **`yum`** *are* case-sensitive.

As with the search and system update functions, **`yum`** begins the process by downloading data files from each of the repositories that it is configured to use. Once **`yum`** has determined the steps to carry out the task you are presented with the proposed package changes, which you can either approve or reject. By default no changes are made to your system unless you approve.

Downloaded Packages

The RPM packages downloaded and used by **`yum`** are held in sub-directories of **`/var/cache/yum/`**, with one sub-directory per repository. You may copy these cached packages and use them elsewhere if you wish. Removing a package from your system does not delete the downloaded RPM from the cache. See *Section 8.2, "Clearing the **`yum`** Caches"* for details on purging the caches.

6.1. Installing New Software with yum

To install or update software, **`yum`** examines the package caches on your system and each of the configured package sources to determine the best set of actions to produce the required result. This may include installing or updating other packages in addition to the package that you specified.

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

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

Enter the root password when prompted.

6.2. Installing Software from a Package with yum

Use the **localinstall** option to install software from an individual package file on your system. In this mode **yum** simply installs the specified package without connecting to any repository. You are responsible for ensuring that all of the dependencies are already installed on your system.

To install the package **tsclient-0.132-4.i386.rpm**, enter the command:

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

Enter the root password when prompted.

Public Key is Required

You must ensure that the public key for the package source has been imported before installing a package without a repository. Refer to *Section 7.2, “Manually Authorizing Package Sources”*

6.3. Updating Software with yum

Updating a software package follows the same process as installing a new package. For example, to update the **tsclient** package to the latest version, type:

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

Enter the root password when prompted.

6.4. Removing Software with yum

To remove software, **yum** examines your system for both the specified software, and any other software that must also be removed in order to safely uninstall it.

To remove the **tsclient** package from your system the full command is:

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

Enter the root password when prompted.

7. Using Other Software Repositories

Projects and individuals that provide RPM packages through **yum** repositories will provide details on their Website. The Fedora Extras\; project is the official source for additional packages.

The Website for Fedora Extras\; is here:

http://fedora.redhat.com/projects/extras/

Repositories for Early Versions of Fedora Core;

Fedora Extras\; does not provides packages for Fedora Core; 2 or earlier. The official Website for additional packages for Fedora Core; 1 and Fedora Core; 2 is: *http://www.fedora.us/*

You should use these sites for software that is not included with Fedora Core;. If these sites do not provide packages for a specific piece of software, the manufacturer of the software may provide or recommend a repository.

7.1. Adding a Repository as a Package Source

Fedora Core; includes a **`yum`** package that has Fedora; repositories in the configuration. 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 Websites.

Definition File Extension

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

Adding a file to the definitions directory requires root access. To copy the definition file **`example.repo`**, type the command:

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

Enter the root password when prompted.

The configuration file for each repository should include the location of the 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"*.

Repositories and up2date Channels

You should also add new package repositories as **up2date** channels to ensure consistency between the behavior of the two applications.

7.2. Manually Authorizing Package Sources

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

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

Enter the root password when prompted.

You may also import public keys directly from a Website. For example, to import the file **GPG-PUB-KEY.asc** on the website *www.therepository.com* the command would be:

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

Public Keys and up2date

The **up2date** utility automatically uses the public key for Fedora Core; packages. It does not add the public key to the keyring that is used by both **yum** and the **rpm** utility.

7.3. Understanding Repository Compatibility

The Fedora Extras\; project provides packages that are built to the same standards as the packages that are part of Fedora Core;. Third-party packages should be compatible with these official packages, unless the provider specifically states otherwise.

You should still always check the Website of the provider for compatibility information before attempting to use a repository. Repositories often provide packages that are specifically intended for use with packages that are supplied by other repositories. In some cases separate third-party repository providers may each offer different versions of the same software, preventing those repositories from being safely used together by your Fedora Core; system.

Packages that have been made for one version of Fedora Core; are usually not compatible with other versions of Fedora Core;. The Website of the provider should specifically state which versions of Fedora Core; they support.

Old Versions of yum and Current Repositories

The data format for repository indexes changed with version 2.11 of **yum**. This was the version supplied with Fedora Core; 3. Repository providers should specify the versions of Fedora Core; that they support. All repositories compatible with current versions of **yum** can also be identified by the fact that they have a sub-directory called **repo-data/**.

8. Maintaining yum

The **yum** system does not require any routine maintenance. It is useful to disable or remove repository definitions that are no longer required, as each repository that is defined and enabled is checked for every operation. You may also wish to periodically remove files relating to unwanted packages, in order to save disk space.

8.1. Disabling or Removing Package Sources

Set **enable=0** in a definition file to prevent **yum** using that repository. Any definition file with this setting is ignored.

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/`**.

If you will not be using any more packages from that source then you should also remove their public key from the **`rpm`** keyring. To remove a public key you first need to know the identification name used by **`rpm`**. You may view the details of all public keys with the command:

```
rpm -qi gpg-pubkey-*
```

The identification name for a key is **`gpg-pubkey-Version_number-Release_number`**. For example, the Fedora Project; public key is currently version 4f2a6fd2, release 3f9d9d3b. The **`rpm`** identification for this key is **`gpg-pubkey-4f2a6fd2-3f9d9d3b`**.

Once you know the identification name of the key, use the command **`rpm -e`** to remove it. To remove the Fedora Project; public key shown above the exact command would be:

```
su -c 'rpm -e gpg-pubkey-4f2a6fd2-3f9d9d3b'
```

Enter the root password when prompted.

8.2. Clearing the yum Caches

By design, **`yum`** does not automatically delete any of the packages or package header files that it downloads, so that these can be reused. Header files accumulate over time, and these may be purged with the command:

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

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

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

In both cases, enter the root password when prompted.

9. Managing yum Repositories

You may wish to create your own software repositories, or maintain a copy of another repository.

Old versions of yum use a different repository utility

These procedures are for repositories that are compatible with version 2.11 of **`yum`** and above. You must use the **`yum-arch`** utility that was included with **`yum`** 2.10 to enable repositories for older versions of **`yum`**.

9.1. Creating a New Repository

A software repository is simply a directory containing package files, with a sub-directory for the package index files used by **`yum`**. Other types of files can be held in the main directory without interfering with use of the repository. The **`data/`** sub-directory and the XML files it contains are created and updated with the **`createrepo`** utility

Creating Repositories Requires an Extra Package

You must install the **`createrepo`** package from Fedora Core; in order to be able to make repositories.

To make a directory into a **`yum`** repository:

1. Copy the RPM packages that you are distributing into the directory.

2. Open a terminal window.

3. In the terminal window type: **`createrepo /path/to/directory`**. Substitute the path to your package directory for **`/path/to/directory/`**.

4. In the terminal window type: **`chmod a+x /path/to/directory/repodata/`**. Substitute the path to your package directory for **`/path/to/directory/`**.

5. Ensure that the directory is available via your chosen network protocols.

The repository is now ready for use.

Create a definition file for this new repository. If you are distributing packages that you have created yourself then you also need to make the GPG public key for your signature available, so that others can verify the packages. The simplest way to make these files available is to put the public key and repository definition files on the same Website or FTP site as the repository.

Creating a definition file is described in *Section 9.2, “Repository Definition Files”*. Packaging building is beyond the scope of this document.

Repositories and Management Utilities

The **`createrepo`** utility makes no changes to the directory other than adding a **`repo-data/`** sub-directory. Adding index files for other utilities does not interfere with **`yum`**.

9.2. Repository Definition Files

Create and edit repository definition files with a text editor. Definition files are plain-text with a standard format:

```
      [serverid]
      name=Some longer name and description for this repository
      baseurl=url://path/to/repository-copy-1/
              url://path/to/repository-copy-2/
      enable=(0 to disable this file, or 1 to enable it)
      gpgcheck=(0 to disable checking signatures of packages from this repository, or 1 to
enable checking)
```

```
        gpgkey=url://path/to/gpg-key-file
```

Example 1. Format of **yum** Repository Definition Files

The **baseurl** must specify the complete URL for the root directory of the repository, including the **http://**, **https://** or **ftp://** prefix. You may also specify a directory on your system, by using the prefix **file://** in the **baseurl**.

Logging in to Protected Repositories

To use a password-protected repository, include the correct username and password in the **baseurl**. For example, *ftp://user:password@myrepository.com/$releasever/mypackages/*.

If possible, list more than one directory or server that holds a copy of the repository. This enables **yum** to use another repository if the first is unavailable. By default **yum** randomly selects repositories from the **baseurl** list. To force **yum** to use them in sequence, add the option **failovermethod=priority**.

It is also good practice to use variables like **$releasever** in the URL, rather than setting these to a specific value. The available variables are listed on the **man** page for **yum.conf**. Using variables enables the same definition to function when your system is upgraded to a later version, or if the configuration is copied to another machine.

A definition file is shown below that uses all of these features. In this example, copies of the repository are held in the directory **/srv/software/Fedora;/3;/mypackages/** on the system itself, in the directory **software/Fedora;/3;/mypackages/** on the Web server *www.my-repository.com/*, and in the directory **pub/software/Fedora;/3;/mypackages/** on the FTP server *server.another-repository.org*. Here, **yum** will access the FTP server with the username **yum-user** and the password **qwerty**. The **failovermethod** ensures that **yum** will check the copy on the local machine, before trying the servers in sequence.

```


        [MyPackages]
        name=Some packages for Fedora $releasever
        baseurl=file:///srv/software/fedora/$releasever/mypackages/
                http://www.my-repository.com/software/fedora/$releasever/mypackages/
                ftp://yum-user:qwerty@anotherserver.another-repository.org/pub/software/fedora/
$releasever/mypackages/
        failovermethod=priority
        enable=1
        gpgcheck=1
        gpgkey=http://www.my-repository.com/software/fedora/keys/RPM-GPG-KEY.asc
```

Example 2. A **yum** Repository Definition File with Failover

To use a list of servers, substitute **mirrorlist** for **baseurl**.

Set **gpgcheck=0** if it is necessary to disable signature checking for the packages provided by this repository. Avoid distributing or installing unsigned packages.

9.3. Updating a Repository

Whenever a package is added, or replaced with a different version, you must run **`createrepo`** again for the index files to be updated. If you are mirroring an existing repository then you may assume that the site administrator updates the indexes, but for safety you should add this to your synchronization scripts. The **`createrepo`** utility can be run as frequently as you wish.

10. Using yum with a Proxy Server

Repositories may be accessed through standard proxy servers. If your system is connected to the Internet through a Web proxy server, specify the details of the server 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.

For example, 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 3. Configuration File Settings for Using A Proxy Server

Global Settings

Defining a proxy server in **`/etc/yum.conf`** means that *all* users connect to the proxy server with those details when using **`yum`**.

To enable proxy access for a specific user, add the lines in the example box below to their 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 4. Profile Settings for Using a Proxy Server

If the proxy server requires a username and password then add these to the URL. For example, to include the username **`yum-user`** and the password **`qwerty`**:

```
# 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 5. Profile Settings for a Secured Proxy Server

`http_proxy` Variable with Other Utilities

The **`http_proxy`** 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.

Index

Index