Red Hat Enterprise Linux 9.0 Beta

Managing software with YUM

A guide to managing software with YUM in Red Hat Enterprise Linux 9

Last Updated: 2021-11-15

Red Hat Enterprise Linux 9.0 Beta Managing software with YUM

A guide to managing software with YUM in Red Hat Enterprise Linux 9

Legal Notice

Abstract

This document describes searching, discovering, installing, and using content in the AppStream and BaseOS repositories using the YUM tool in Red Hat Enterprise Linux 9. This includes a description of how to use modules, application streams, and profiles.

Table of Contents

RHEL BETA RELEASE

Red Hat provides Red Hat Enterprise Linux Beta access to all subscribed Red Hat accounts. The purpose of Beta access is to:

- Provide an opportunity to customers to test major features and capabilities prior to the general availability release and provide feedback or report issues.
- Provide Beta product documentation as a preview. Beta product documentation is under development and is subject to substantial change.

Note that Red Hat does not support the usage of RHEL Beta releases in production use cases. For more information, see What does Beta mean in Red Hat Enterprise Linux and can I upgrade a RHEL Beta installation to a General Availability (GA) release?.

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

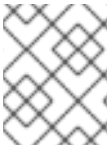
We appreciate your input on our documentation. Please let us know how we could make it better. To do so:

- For simple comments on specific passages:
  1. Make sure you are viewing the documentation in the *Multi-page HTML* format. In addition, ensure you see the **Feedback** button in the upper right corner of the document.
  2. Use your mouse cursor to highlight the part of text that you want to comment on.
  3. Click the **Add Feedback** pop-up that appears below the highlighted text.
  4. Follow the displayed instructions.
- For submitting more complex feedback, create a Bugzilla ticket:
  1. Go to the Bugzilla website.
  2. As the Component, use **Documentation**.
  3. Fill in the **Description** field with your suggestion for improvement. Include a link to the relevant part(s) of documentation.
  4. Click **Submit Bug**.

CHAPTER 1. SOFTWARE MANAGEMENT TOOLS IN RED HAT ENTERPRISE LINUX 9

Throughout this document, **YUM** and **DNF** can be used interchangeably.

In Red Hat Enterprise Linux 9, software installation is ensured by the **DNF** tool. Red Hat continues to support the usage of the **yum** term for consistency with previous major versions of RHEL. If you type **dnf** instead of **yum**, the command works as expected because both are aliases for compatibility.



NOTE

Although RHEL 8 and RHEL 9 are based on **DNF**, they are compatible with **YUM** used in RHEL 7.

CHAPTER 2. DISTRIBUTION OF CONTENT IN RHEL 9

The following sections provide an overview of software distribution in Red Hat Enterprise Linux 9:

- Section 2.1. “Repositories” describes how content in Red Hat Enterprise Linux 9 is split into BaseOS and AppStream.
- Section 2.2. “Application Streams” describes the concept of Application Streams.
- Section 2.3. “Modules” describes the concept of modules.
- Section 2.4. “Module streams” describes the organization of content by version.
- Section 2.5. “Module profiles” describes the organization of content by purpose.

2.1. REPOSITORIES

RHEL 9 content is distributed through the two main repositories: BaseOS and AppStream. Both the BaseOS and AppStream content sets are required for a basic RHEL installation and are available with all RHEL subscriptions. For installation instructions, see the Performing a standard RHEL installation document.

BaseOS

Content in the BaseOS repository is intended to provide the core set of the underlying OS functionality that provides the foundation for all installations. This content is available in the RPM format and is subject to support terms similar to those in previous releases of Red Hat Enterprise Linux.

AppStream

Content in the AppStream repository includes additional user-space applications, runtime languages, and databases in support of the varied workloads and use cases.

CodeReady Linux Builder

The CodeReady Linux Builder repository is available with all RHEL subscriptions. It provides additional packages for use by developers. Packages included in the CodeReady Linux Builder repository are unsupported.

Additional resources

- Performing a standard RHEL installation

2.2. APPLICATION STREAMS

Multiple versions of user-space components are delivered as Application Streams and updated more frequently than the core operating system packages. This provides greater flexibility to customize RHEL without impacting the underlying stability of the platform or specific deployments.

Each Application Stream component has a given lifecycle, either the same as RHEL 9 or shorter, more suitable to the particular application.

Application Streams are available in the familiar RPM format, as an extension to the RPM format called modules, as Software Collections, or as Flatpaks.

RHEL 9 improves Application Streams experience by providing initial Application Stream versions that can be simply installed as RPM packages by using the traditional **yum install** command.

Some additional Application Stream versions will be distributed as modules with a shorter lifecycle in future minor RHEL 9 releases.

Additional resources

- Red Hat Enterprise Linux Life Cycle

2.3. MODULES

A module is a set of RPM packages that represent a component and are usually installed together. A typical module contains packages with an application, packages with the application-specific dependency libraries, packages with documentation for the application, and packages with helper utilities.

2.4. MODULE STREAMS

Module streams are filters that can be imagined as virtual repositories in the AppStream physical repository. Module streams represent versions of the AppStream components. Each of the streams receives updates independently.

Module streams can be active or inactive. Active streams give the system access to the RPM packages within the particular module stream, allowing installation of the respective component version. Streams are active if they are explicitly enabled by a user action.

Only one stream of a particular module can be active at a given point in time. Thus only one version of a component can be installed on a system. Different versions can be used in separate containers.

Each module can have a default stream. Default streams make it easy to consume RHEL packages the usual way without the need to learn about modules. The default stream is active, unless the whole module has been disabled or another stream of that module enabled.

Certain module streams depend on other module streams. For example, the **perl-App-cpanminus**, **perl-DBD-MySQL**, **perl-DBD-Pg**, **perl-DBD-SQLite**, **perl-DBI**, **perl-YAML**, and **freeradius** module streams depend on certain **perl** module streams.

To select a particular stream for a runtime user application or a developer application, consider the following:

- Required functionality and which component versions support that functionality
- Compatibility

For per-component changes, see the Release Notes.

Additional resources

- Modular dependencies and stream changes

2.5. MODULE PROFILES

A profile is a list of recommended packages to be installed together for a particular use case such as for a server, client, development, minimal install, or other. These package lists can contain packages outside the module stream, usually from the BaseOS repository or the dependencies of the stream.

Installing packages by using a profile is a one-time action provided for the user's convenience. It does

not prevent installing or uninstalling any of the packages provided by the module. It is also possible to install packages by using multiple profiles of the same module stream without any further preparatory steps.

Each module stream can have any number of profiles, including none. For any given module stream, some of its profiles can be marked as *default* and are then used for profile installation actions when no profile is explicitly specified. However, existence of a default profile for a module stream is not required.

CHAPTER 3. CONFIGURING YUM

The configuration information for **YUM** and related utilities is stored in the **/etc/yum.conf** file. This file contains one mandatory **[main]** section, which enables you to set **YUM** options that have global effect.

The following sections describe how to:

- View the current **YUM** configurations.
- Set **YUM [main]** options.
- Use **YUM** plug-ins.

3.1. VIEWING THE CURRENT YUM CONFIGURATIONS

The following procedure describes how to display the current **YUM** configuration.

Procedure

- To display the current values of global yum options specified in the **[main]** section of the **/etc/yum.conf** file, use:

```
# yum config-manager --dump
```

3.2. SETTING YUM MAIN OPTIONS

The **/etc/yum.conf** configuration file contains one **[main]** section. The key-value pairs in this section affect how **YUM** operates and treats repositories.

You can add additional options under the **[main]** section heading in **/etc/yum.conf**.

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

3.3. USING YUM PLUG-INS

YUM provides plug-ins that extend and enhance its operations. Certain plug-ins are installed by default.

The following procedures describe how to enable, configure, and disable **YUM** plug-ins.

3.3.1. Managing YUM plug-ins

The plug-in configuration files always contain a **[main]** section in which the **enabled=** option controls whether the plug-in is enabled when you run **yum** commands. If this option is missing, you can add it manually to the file.

Every installed plug-in has its own configuration file in the **/etc/dnf/plugins/** directory. You can enable or disable plug-in specific options in these files.

3.3.2. Enabling and disabling YUM plug-ins

In the **YUM v4** version of the **YUM** tool, plug-ins are enabled by default.

The following procedure describes how to disable or enable all **YUM** plug-ins, disable all plug-ins for a particular command, or certain **YUM** plug-ins for a single command.

Procedure

- To disable or enable all **YUM** plug-ins, ensure a line beginning with **plugins=** is present in the **[main]** section of the **/etc/yum.conf** file.

  1. To disable all **YUM** plug-ins, set the value of **plugins=** to **0**.

  2. To enable all **YUM** plug-ins, set the value of **plugins=** to **1**.

IMPORTANT

Disabling all plug-ins is **not** advised. Certain plug-ins provide important **YUM** services and commands. In particular, the **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 advisable only when diagnosing a potential problem with **YUM**.

- To disable all **YUM** plug-ins for a particular command, append **--noplugins** option to the command. For example, to disable **YUM** plug-ins for the **update** command:

```
# yum --noplugins update
```

- To disable certain **YUM** plug-ins for a single command, append **--disableplugin=*plugin-name*** option to the command. For example, to disable certain yum plug-ins for the **update** command:

```
# yum update --disableplugin=plugin-name
```

  Replace *plugin-name* with the name of the plug-in.

- To enable certain **YUM** plug-ins for a single command, append **--enableplugin=*plugin-name*** option to the command. For example, to disable certain yum plug-ins for the **update** command:

```
# yum update --enableplugin=plugin-name
```

  Replace *plugin-name* with the name of the plug-in.

CHAPTER 4. SEARCHING FOR RHEL 9 CONTENT

The following sections describe how to locate and examine content in the AppStream and BaseOS repositories in Red Hat Enterprise Linux 9:

- Section 4.1. "Searching for software packages" describes how to search for packages providing desired content.
- Section 4.2. "Listing software packages" describes how to list installed and available packages.
- Section 4.3. "Listing repositories" describes how to list enabled and disabled repositories.
- Section 4.4. "Displaying package information" describes how to display information about available packages.
- Section 4.5. "Listing package groups" describes how to list installed and available package groups.
- Section 4.6. "Listing available modules" describes how to list available modules and find out details about them.

4.1. SEARCHING FOR SOFTWARE PACKAGES

This section describes steps needed for finding a package providing a particular application or other content.

Procedure

- To search for a package, use:

  ```
  # yum search term
  ```

  Replace *term* with a term related to the package.

  Note that the **yum search** command returns term matches within the name and summary of the packages. This makes the search faster and enables you to search for packages you do not know the name of, but for which you know a related term.

- To include term matches within package descriptions, use:

  ```
  # yum search --all term
  ```

  Replace *term* with a term you want to search for in a package name, summary, or description.

  Note that the **yum search --all** command enables a more exhaustive but slower search.

4.2. LISTING SOFTWARE PACKAGES

The following procedure describes how to list available packages with **yum**.

Procedure

- To list information on all installed and available packages, use:

```
# yum list --all
```

- To list all packages installed on your system, use:

```
# yum list --installed
```

Alternatively:

```
# yum repoquery --installed
```

- To list all packages in all enabled repositories that are available to install, use:

```
# yum list --available
```

Alternatively:

```
# yum repoquery
```

Note that you can filter the results by appending global expressions as arguments. For more details, see Specifying global expressions in yum input .

4.3. LISTING REPOSITORIES

The following procedure describes how to list repositories with **yum**.

Procedure

- To list all enabled repositories on your system, use:

```
# yum repolist
```

- To list all disabled repositories on your system, use:

```
# yum repolist --disabled
```

- To list both enabled and disabled repositories, use:

```
# yum repolist --all
```

- To list additional information about the repositories, use:

```
# yum repoinfo
```

Note that you can filter the results by passing the ID or name of repositories as arguments or by appending global expressions. For more details, see Specifying global expressions in yum input .

4.4. DISPLAYING PACKAGE INFORMATION

The following procedure describes how to display package information using **yum**.

Procedure

- To display information about one or more available packages, use:

  ```
  # yum info package-name
  ```

  Replace *package-name* with the name of the package.

  Alternatively:

  ```
  # yum repoquery --info package-name
  ```

  Replace *package-name* with the name of the package.

- To display information about one or more packages installed on your system, use:

  ```
  # yum repoquery --info --installed package-name
  ```

  Replace *package-name* with the name of the package.

Note that you can filter the results by appending global expressions as arguments. For more details, see Specifying global expressions in yum input .

4.5. LISTING PACKAGE GROUPS

The following procedure describes how to list package groups using **yum**.

- To view the number of installed and available groups, use:

  ```
  # yum group summary
  ```

- To list all installed and available groups, use:

  ```
  # yum group list
  ```

  Note that you can filter the results by appending command line options for the **yum group list** command (**--hidden**, **--available**). For more available options see the man pages.

- To list mandatory and optional packages contained in a particular group, use:

  ```
  # yum group info group-name
  ```

  Replace *group-name* with the name of the group.

Note that you can filter the results by appending global expressions as arguments. For more details, see Specifying global expressions in yum input .

4.6. LISTING AVAILABLE MODULES

The following procedure describes how to find what modules are available and what their details are using **yum**.

Procedure

- To list module streams available to your system:

```
# yum module list
```

  The output of this command lists module streams with name, stream, profiles, and summary on a separate line.

- To display details about a module, including a description, a list of all profiles, and a list of all provided packages:

```
# yum module info module-name
```

- To list which of these packages are installed by each of module profiles:

```
# yum module info --profile module-name
```

- To display the current status of a module, including enabled streams and installed profiles:

```
# yum module list module-name
```

- To find out which modules provide a specific package:

```
# yum module provides package
```

  If the package is available outside any modules, the output of this command is empty.

Additional resources

- Modules
- Module streams
- Module profiles

4.7. SPECIFYING GLOBAL EXPRESSIONS IN YUM INPUT

yum commands allow you to filter the results by appending one or more *global expressions* as arguments. Global expressions must be escaped when passed as arguments to the **yum** command.

The following procedure describes two ways to ensure global expressions are passed to **yum** as intended.

Procedure

- Double-quote or single-quote the entire global expression:

```
# yum provides "*/file-name"
```

  Replace *file-name* with the name of the file.

  Note that the *file-name* must be preceeded either by / or */ character sequence to provide the desired outcome.

- Escape the wildcard characters by preceding them with a backslash (\) character:

```
# yum provides \*/file-name
```

  Replace *file-name* with the name of the file.

4.8. ADDITIONAL RESOURCES

- Commands for listing content in RHEL 9

CHAPTER 5. INSTALLING RHEL 9 CONTENT

The following sections describe how to install content in Red Hat Enterprise Linux 9:

- Section 5.1. "Installing packages" includes steps for installing a package.
- Section 5.2. "Installing package groups" describes how to install a package group.
- Section 5.3. "Installing modular content" describes steps to install sets of packages provided by modules, streams, and profiles.
- Section 5.4. "Running installed content" provides details for running RHEL 9 installed content.
- Section 5.5. "Setting module default streams" describes how to override repository module defaults on a per-system basis

5.1. INSTALLING PACKAGES

The following procedure describes how to install packages using **yum**.

Procedure

- Install the package:

  ```
  # yum install package-name
  ```

  Replace *package-name* with the name of the package.

  - If the package is not provided by any module stream, this procedure is identical to the procedure used on previous versions of Red Hat Enterprise Linux.
  - If the package is provided by a module stream that is enabled, the package is installed without any further manipulation.
  - If the package is provided by a module stream marked as default, the **YUM** tool automatically transparently enables that module stream before installing this package.
  - If the package is provided by a module stream that is not active (neither of the above cases), it is not recognized until you manually enable the respective module stream.

- To install multiple packages and their dependencies simultaneously, use:

  ```
  # yum install package-name-1 package-name-2
  ```

  Replace *package-name-1* and *package-name-2* with the names of the packages.

- When installing packages on a *multilib* system (AMD64, Intel 64 machine), you can specify the architecture of the package by appending it to the package name:

  ```
  # yum install package-name.arch
  ```

  Replace *package-name.arch* with the name and architecture of the package.

- If you know the name of the binary you want to install, but not the package name, you can use the path to the binary as an argument:

```
# yum install /usr/sbin/binary-file
```

Replace */usr/sbin/binary-file* with a path to the binary file.

yum searches through the package lists, finds the package which provides */usr/sbin/binary-file*, and prompts you as to whether you want to install it.

- To install a previously-downloaded package from a local directory, use:

```
# yum install /path/
```

Replace */path/* with the path to the package.

Note that you can optimize the package search by explicitly defining how to parse the argument. See **TBD** xref:[Specifying a package name in yum input] for more details.

Additional resources

- Installing modular content

5.2. INSTALLING PACKAGE GROUPS

The following procedure describes how to install a package group by a group name or by a groupID using **yum**.

Procedure

- To install a package group by a group name, use:

```
# yum group install group-name
```

Replace *group-name* with the full name of the group or environmental group.

- To install a package group by a groupID, use:

```
# yum group install groupID
```

Replace *groupID* with the ID of the group.

5.3. INSTALLING MODULAR CONTENT

This section describes how to install modular content provided by a module stream or a profile.

Default module streams ensure that users can install packages without caring about the modular features. If you want to install a different version of a package from a non-default stream, you must enable the stream before installing the package.

Some modules do not define default streams. In such a case, you must also enable the stream.

Prerequisites

- You understand the concept of an *active module stream*.
- You do not have any packages installed from another stream of the same module.

Procedure

- To enable a module stream:

  ```
  # yum module enable module-name:stream
  ```

  Replace *module-name* and *stream* with names of the module and stream.

  yum asks for confirmation and the stream is enabled and active. If another stream of the module was previously active because it was default, it is no longer active.

- Install an active module stream (the one you have enabled):

  ```
  # yum module install module-name
  ```

- Install a selected module stream:

  ```
  # yum module install module-name:stream
  ```

  The selected stream is automatically enabled. If a default profile is defined for the stream, this profile is automatically installed.

- Install a selected profile of the module stream:

  ```
  # yum module install module-name:stream/profile
  ```

  This enables the stream and installs the recommended set of packages for a given stream (version) and profile (purpose) of the module.

Additional resources

- Modules
- Module streams
- Module profiles

5.4. RUNNING INSTALLED CONTENT

New commands are usually enabled after you install content from RHEL 9 repositories. If the commands originated from an RPM package or RPM packages were enabled by a module, the experience of using the command should be no different.

Procedure

- To run the new commands use them directly:

  ```
  $ command
  ```

5.5. SETTING MODULE DEFAULT STREAMS

In RHEL 9, no module default streams are defined in the repository that contains the modules. You can set a default stream by creating a configuration file in the **/etc/dnf/modules.defaults.d/** directory.

The following procedure describes how to set the default stream via the **/etc/dnf/modules.defaults.d/** directory.

Prerequisites

- You understand the concept of an *active module stream*.

Procedure

1. Create a YAML configuration file in the **/etc/dnf/modules.defaults.d/** drop-in directory.

```
---
document: modulemd-defaults
version: 1
data:
  module: <module>
  stream: "<stream>"
  profiles:
    <profile>
    <profile>
…
```

   The output above represents the default definition present for the **<module>** module at the time of this writing.

2. To set the default stream to 13, the following configuration can be implemented:

```
---
document: modulemd-defaults
version: 1
data:
  module: <module>
  stream: "13"
  profiles:
    <profile>
    <profile>
…
```

5.6. ADDITIONAL RESOURCES

- **yum(8)** man page
- Commands for installing content in RHEL 9

CHAPTER 6. UPDATING RHEL 9 CONTENT

YUM allows you to check if your system has any pending updates. You can list packages that need updating and choose to update a single package, multiple packages, or all packages at once. If any of the packages you choose to update have dependencies, they are updated as well.

The following sections describe how to use **YUM** to update content in Red Hat Enterprise Linux 9:

- Section 6.1. "Checking for updates" describes how to check the available updates.
- Section 6.2. "Updating packages" describes how to update a single package, package group, or all packages and their dependencies.
- Section 6.3. "Updating security-related packages" describes how to apply security updates.

6.1. CHECKING FOR UPDATES

The following procedure describes how to check the available updates for packages installed on your system using **yum**.

Procedure

- To see which packages installed on your system have available updates, use:

  ```
  # yum check-update
  ```

  The output returns the list of packages and their dependencies that have an update available.

6.2. UPDATING PACKAGES

The following procedure describes how to update a single package, a package group, or all packages and their dependencies using **yum**.

Procedure

- To update all packages and their dependencies, use:

  ```
  # yum update
  ```

- To update a single package, use:

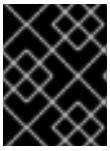
  ```
  # yum update package-name
  ```

  Replace *package-name* with the name of the package.

- To update a package group, use:

  ```
  # yum group update group-name
  ```

  Replace *group-name* with the name of the package group.



IMPORTANT

When applying updates to the kernel, **yum** always installs a new kernel regardless of whether you are using the yum update or yum install command.

6.3. UPDATING SECURITY-RELATED PACKAGES

The following procedure describes how to update security-related packages using **yum**.

Procedure

- To upgrade to the latest available packages that have security errata, use:

```
# yum update --security
```

- To upgrade to the last security errata packages, use:

```
# yum update-minimal --security
```

CHAPTER 7. AUTOMATING SOFTWARE UPDATES IN RHEL 9

To check and download package updates automatically and regularly, you can use the **DNF Automatic** tool that is provided by the **dnf-automatic** package.

DNF Automatic is an alternative command-line interface to **YUM** that is suited for automatic and regular execution using systemd timers, cron jobs, and other such tools.

DNF Automatic synchronizes package metadata as needed, checks for updates available, and then performs one of the following actions depending on how you configure the tool:

- Exit
- Download updated packages
- Download and apply the updates

The outcome of the operation is then reported by a selected mechanism, such as the standard output or email.

The following sections describe how to automate software updates in Red Hat Enterprise Linux 9:

- Section 7.1. "Installing DNF Automatic" describes how to install the **DNF Automatic** tool.
- Section 7.2. "DNF Automatic configuration file" describes the **DNF Automatic** configuration file and its sections.
- Section 7.3. "Enabling DNF Automatic" describes how to enable the **DNF Automatic** tool.
- Section 7.4. "Overview of the systemd timer units included in the **dnf-automatic** package" lists the **dnf-automatic** systemd timer units.

7.1. INSTALLING DFN AUTOMATIC

The following procedure describes how to install the **DNF Automatic** tool.

Procedure

- To install the **dnf-automatic** package, use:

```
# yum install dnf-automatic
```

Verification

- To verify the successful installation, confirm the presence of the **dnf-automatic** package by running the following command:

```
# rpm -qi dnf-automatic
```

7.2. DNF AUTOMATIC CONFIGURATION FILE

By default, **DNF Automatic** uses **/etc/dnf/automatic.conf** as its configuration file to define its behavior.

The configuration file is separated into the following topical sections:

- **[commands]** section
  Sets the mode of operation of **DNF Automatic**.
- **[emitters]** section
  Defines how the results of **DNF Automatic** are reported.
- **[command_email]** section
  Provides the email emitter configuration for an external command used to send email.
- **[email]** section
  Provides the email emitter configuration.
- **[base]** section
  Overrides settings from the main configuration file of **yum**.

With the default settings of the **/etc/dnf/automatic.conf** file, **DNF Automatic** checks for available updates, downloads them, and reports the results as standard output.

WARNING

Settings of the operation mode from the **[commands]** section are overridden by settings used by a systemd timer unit for all timer units except **dnf-automatic.timer**.

Additional resources

- DNF Automatic documentation
- **man dnf-automatic** man page
- Overview of the systemd timer units included in the dnf-automatic package

7.3. ENABLING DNF AUTOMATIC

To run **DNF Automatic**, you always need to enable and start a specific systemd timer unit. You can use one of the timer units provided in the **dnf-automatic** package, or you can write your own timer unit depending on your needs.

The following procedure describes how to enable **DNF Automatic**.

Prerequisites

- You specified the behavior of **DNF Automatic** by modifying the **/etc/dnf/automatic.conf** configuration file.

Procedure

- To select, enable, and start a systemd timer unit that **downloads** available updates, use:

```
# systemctl enable dnf-automatic-download.timer

# systemctl start dnf-automatic-download.timer
```

- To select, enable, and start a systemd timer unit that **downloads and installs** available updates, use:

```
# systemctl enable dnf-automatic-install.timer

# systemctl start dnf-automatic-install.timer
```

- To select, enable, and start a systemd timer unit that **reports** available updates, use:

```
# systemctl enable dnf-automatic-notifyonly.timer

# systemctl start dnf-automatic-notifyonly.timer
```

- To select, enable, and start a systemd timer unit that behaves according to settings in the **/etc/dnf/automatic.conf** configuration file, use:

```
# systemctl enable dnf-automatic.timer

# systemctl start dnf-automatic.timer
```

  The default behavior of this timer unit is similar to **dnf-automatic-download.timer**: it downloads the updated packages but does not install them.

- Optionally, select, enable, and start a systemd timer unit in one command using the **--now** option. For example:

```
# systemctl enable --now dnf-automatic-download.timer
```

NOTE

You can also run **DNF Automatic** by executing the **/usr/bin/dnf-automatic** file directly from the command line or from a custom script.

Verification

- To verify that the timer is enabled, run the following command:

```
# systemctl status <systemd timer unit>
```

Additional resources

- **man dnf-automatic** man page
- Overview of the systemd timer units included in the dnf-automatic package
- DNF Automatic configuration file

7.4. OVERVIEW OF THE SYSTEMD TIMER UNITS INCLUDED IN THE DNF-AUTOMATIC PACKAGE

The systemd timer units take precedence and override the settings in the **/etc/dnf/automatic.conf** configuration file when downloading and applying updates.

For example if you set:

```
download_updates = yes
```

in the **/etc/dnf/automatic.conf** configuration file, but you have activated the **dnf-automatic-notifyonly.timer unit**, the packages will not be downloaded.

The **dnf-automatic** package includes the following systemd timer units:

Table 7.1. systemd timers included in the dnf-automatic package

Timer unit	Function	Overrides settings in the /etc/dnf/automatic.conf file?
dnf-automatic-download.timer	Downloads packages to cache and makes them available for updating. Note: This timer unit does not install the updated packages. To perform the installation, you must execute the **dnf update** command.	Yes
dnf-automatic-install.timer	Downloads and installs updated packages.	Yes
dnf-automatic-notifyonly.timer	Downloads only repository data to keep the repository cache up-to-date and notifies you about available updates. Note: This timer unit does not download or install the updated packages	Yes

Timer unit	Function	Overrides settings in the **/etc/dnf/automatic.conf** file?
dnf-automatic.timer	The behavior of this timer when downloading and applying updates is specified by the settings in the **/etc/dnf/automatic.conf** configuration file. Default behavior is the same as for the **dnf-automatic-download.timer** unit: it downloads packages, but does not install them.	No

Additional resources

- **man dnf-automatic manual** man page
- DNF Automatic configuration file

CHAPTER 8. REMOVING RHEL 9 CONTENT

The following sections describe how to remove content in Red Hat Enterprise Linux 9:

- Section 8.1. "Removing installed packages" describes removing a package.
- Section 8.2. "Removing package groups" describes removing a package group.
- Section 8.3. "Removing installed modular content" describes removing content installed from a module stream or a profile.
  - Section 8.3.1. "Removing packages from an installed profile" describes removing all packages that belong to a selected profile.
  - Section 8.3.2."Removing all packages from a module stream" describes removing all packages installed from a selected module stream.

8.1. REMOVING INSTALLED PACKAGES

The following procedure describes how to remove packages using **yum**.

Procedure

- To remove a particular package and all unused dependent packages, use:

  ```
  # yum remove package-name
  ```

  Replace *package-name* with the name of the package.

  Note that the package is removed together with any other dependent packages.

- To remove multiple packages and their unused dependencies simultaneously, use:

  ```
  # yum remove package-name-1 package-name-2
  ```

  Replace *package-name-1* and *package-name-2* with the names of the packages.

NOTE

yum is not able to remove a package without removing dependent packages.

8.2. REMOVING PACKAGE GROUPS

The following procedure describes how to remove a package either by the group name or the groupID.

Procedure

- To remove a package group by the group name, use:

  ```
  # yum group remove group-name
  ```

  Replace *group-name* with the full name of the group.

- To remove a package group by the groupID, use:

```
# yum group remove groupID
```

Replace *groupID* with the ID of the group.

8.3. REMOVING INSTALLED MODULAR CONTENT

When removing installed modular content, you can remove packages from either a selected profile or the whole stream.

IMPORTANT

YUM will try to remove all packages with a name corresponding to the packages installed with a profile or a stream, including their dependent packages. Always check the list of packages to be removed before you proceed, especially if you have enabled custom repositories on your system.

8.3.1. Removing packages from an installed profile

When you remove packages installed with a profile, all packages with a name corresponding to the packages installed by the profile are removed. This includes their dependencies, with the exception of packages required by a different profile.

Prerequisites

- The selected profile has been installed using the **yum module install *module-name:stream/profile*** command or as a default profile using the **yum install *module-name:stream command***.
- You understand modular dependency resolution.

Procedure

1. Uninstall packages belonging to the selected profile:

```
# yum module remove module-name:stream/profile
```

Replace *module-name*, *stream*, and *profile* with the module, stream, and profile you wish to uninstall.

Alternatively, uninstall packages from all installed profiles within a stream:

```
# yum module remove module-name:stream
```

These operations will not remove packages from the stream that do not belong to any of the profiles.

2. Check the list of packages under **Removing:** and **Removing unused dependencies:** before you proceed with the removal transaction.

To remove all packages from a selected stream, follow instructions in Section 8.3.2 "Removing all packages from a module stream".

8.3.2. Removing all packages from a module stream

When you remove packages installed with a module stream, all packages with a name corresponding to the packages installed by the stream are removed. This includes their dependencies, with the exception of packages required by other modules.

Prerequisites

- The module stream has been enabled and at least some packages from the stream have been installed.
- You understand modular dependency resolution.

Procedure

1. Remove all packages from a selected stream:

   ```
   # yum module remove --all module-name:stream
   ```

   Replace *module-name* and *stream* with the module and stream you wish to uninstall.

2. Check the list of packages under **Removing:** and **Removing unused dependencies:** before you proceed with the removal transaction.
3. Optionally, reset or disable the stream.

If you want to remove only packages from a selected profile, follow instructions in Section 8.3.1. "Removing packages from an installed profile".

Additional resources

- Resetting module streams
- Disabling all streams of a module

8.4. ADDITIONAL RESOURCES

- Commands for removing content in RHEL 9

CHAPTER 9. HANDLING PACKAGE MANAGEMENT HISTORY

The **yum history** command allows you to review the following information

- Timeline of **YUM** transactions
- Dates and times the transactions occurred
- Number of packages affected by the transactions
- Whether the transactions succeeded or were aborted
- If the RPM database was changed between transactions

The **yum history** command can also be used to undo or redo the transactions.

The following section describes how to use **yum** to handle package management history in Red Hat Enterprise Linux 9:

- Section 9.1. "Listing transactions" describes how to list the latest transactions, the latest operations for a selected package, and details of a particular transaction.
- Section 9.2. "Reverting transactions" describes how to revert selected or last transactions.
- Section 9.3. "Repeating transactions" describes how to repeat selected or last transactions.

9.1. LISTING TRANSACTIONS

The following procedure describes how to list the latest **YUM** transactions, the latest operations for a selected package, and details of a particular transaction.

Procedure

- To display a list of all the latest **YUM** transactions, use:

```
# yum history
```

- To display a list of all the latest operations for a selected package, use:

```
# yum history list package-name
```

  Replace *package-name* with the name of the package. You can filter the command output by appending global expressions. For more details, see Specifying global expressions in yum input .

- To display details of a particular transaction, use:

```
# yum history info transactionID
```

  Replace *transactionID* with the ID of the transaction.

9.2. REVERTING TRANSACTIONS

The following procedure describes how to revert a selected transaction or the last transaction using **yum**.

Procedure

- To revert a particular transaction, use:

  ```
  # yum history undo transactionID
  ```

  Replace *transactionID* with the ID of the transaction.

- To revert the last transaction, use:

  ```
  # yum history undo last
  ```

Note that the **yum history undo** command only reverts the steps that were performed during the transaction. If the transaction installed a new package, **yum history undo** uninstalls it. If the transaction uninstalled a package, **yum history undo** reinstalls it. The **yum history undo** command also attempts to downgrade all updated packages to their previous versions, if the older packages are still available.

9.3. REPEATING TRANSACTIONS

The following procedure describes how to repeat a selected transaction or the last transaction using **yum**.

Procedure

- To repeat a particular transaction, use:

  ```
  # yum history redo transactionID
  ```

  Replace *transactionID* with the ID of the transaction.

- To repeat the last transaction, use:

  ```
  # yum history redo last
  ```

Note that the **yum history redo** command only repeats the steps that were performed during the transaction.

CHAPTER 10. MANAGING CUSTOM SOFTWARE REPOSITORIES

The configuration information for **YUM** and related utilities are stored in the **/etc/yum.conf** file. This file contains one or more **[repository]** sections, which allow you to set repository-specific options.

It is recommended to define individual repositories in new or existing **.repo** files in the **/etc/yum.repos.d/** directory.

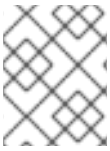
Note that the values you define in individual **[repository]** sections of the **/etc/yum.conf** file override values set in the **[main]** section.

The following sections describe how to manage custom software repositories in Red Hat Enterprise Linux 9:

- Section 10.1. "Setting yum repository options" describes how to set **[repository]** options.
- Section 10.2. "Adding a yum repository" describes how to define a new **yum** repository.
- Section 10.3. "Enabling a yum repository" describes how to enable a **yum** repository added to your system.
- Section 10.4. "Disabling a yum repository" describes how to disable a **yum** repository added to your system.

10.1. SETTING YUM REPOSITORY OPTIONS

The **/etc/yum.conf** configuration file contains the **[*repository*]** sections, where *repository* is a unique repository ID. The **[*repository*]** sections allow you to define individual **YUM** repositories.



NOTE

Do not give custom repositories names used by the Red Hat repositories to avoid conflicts.

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

10.2. ADDING A YUM REPOSITORY

To define a new repository, you can either:

- Add a **[*repository*]** section to the **/etc/yum.conf** file.
- Add a **[*repository*]** section to a **.repo** file in the **/etc/yum.repos.d/** directory. **YUM** repositories commonly provide their own **.repo** file.

NOTE

It is recommended to define your repositories in a **.repo** file instead of **/etc/yum.conf** as all files with the **.repo** file extension in this directory are read by **yum**.

The following procedure describes how to add a **yum** repository to your system.

Procedure

- To add a repository to your system, use:

  ```
  # yum config-manager --add-repo repository_URL
  ```

  Replace *repository_url* with URL pointing to the repository.

WARNING

Obtaining and installing software packages from unverified or untrusted sources other than Red Hat certificate-based **Content Delivery Network** (CDN) constitutes a potential security risk, and could lead to security, stability, compatibility, and maintainability issues.

10.3. ENABLING A YUM REPOSITORY

The following procedure describes how to enable a **yum** repository added to your system.

Procedure

- To enable a repository, use:

  ```
  # yum-config-manager --enable repositoryID
  ```

  Replace *repositoryID* with the unique repository ID.

To list available repository IDs, see Listing software packages.

10.4. DISABLING A YUM REPOSITORY

The following procedure describes how to disable a **yum** repository added to your system.

Procedure

- To disable a yum repository, use:

  ```
  # yum-config-manager --disable repositoryID
  ```

  Replace *repositoryID* with the unique repository ID.

To list available repository IDs, see Listing software packages.

CHAPTER 11. MANAGING VERSIONS OF APPLICATION STREAM CONTENT

Content in the AppStream repository can be available in multiple versions, corresponding to module streams. This chapter describes the operations you need to perform when changing the enabled module streams in other ways than only enabling new module streams.

- Section 11.1. "Modular dependencies and stream changes" describes modular dependency rules.
- Section 11.2. "Interaction of modular and non-modular dependencies" provides details for how the dependencies of module streams affect handling of package dependencies.
- Section 11.3. "Resetting module streams" provides steps for resetting modules to their initial state.
- Section 11.4. "Disabling all streams of a module" provides steps for completely disabling a module and all its streams.

11.1. MODULAR DEPENDENCIES AND STREAM CHANGES

Traditionally, packages providing content depend on further packages, and usually specify the desired dependency versions. For packages contained in modules, this mechanism applies as well, but the grouping of packages and their particular versions into modules and streams provides further constraints. Additionally, module streams can declare dependencies on streams of other modules, independent of the packages contained and provided by them.

After any operations with packages or modules, the whole dependency tree of all underlying installed packages must satisfy all the conditions that the packages declare. Additionally, all module stream dependencies must be satisfied.

As a result:

- Enabling a module stream can require enabling further module streams.
- Installing a module stream profile or installing packages from a stream can require enabling further module streams and installing further packages.
- Disabling a module stream can require disabling other module streams. No packages will be removed automatically.
- Removing a package can require removing further packages. If these packages were provided by modules, the module streams remain enabled in preparation for further installation, even if no packages from these streams are installed any more. This mirrors the behavior of an unused yum repository.

11.2. INTERACTION OF MODULAR AND NON-MODULAR DEPENDENCIES

Modular dependencies are an additional layer on top of regular RPM dependencies. Modular dependencies behave similarly to hypothetical dependencies between repositories. This means that installing different packages requires resolution of both the RPM dependencies and the modular dependencies.

The system will always retain the module and stream choices, unless explicitly instructed to change them. A modular package will receive updates contained in the currently enabled stream of the module that provides this package, but will not upgrade to a version contained in a different stream.

11.3. RESETTING MODULE STREAMS

Resetting a module is an action that returns all of its streams to their initial state – neither enabled nor disabled. If the module has a default stream, that stream becomes active as a result of resetting the module.

The following procedure describes how to reset a module stream to the initial state using **yum**.

Procedure

- Reset the module state:

```
# yum module reset module-name
```

  The module is returned to the initial state. Information about an enabled stream and installed profiles is erased but no installed content is removed.

11.4. DISABLING ALL STREAMS OF A MODULE

Modules that have a default stream will always have one stream active. In situations where the content from all the module streams should not be accessible, it is possible to disable the whole module.

The following procedure describes how to disable all streams of a module using **yum**.

Prerequisites

- You understand the concept of an *active module stream*.

Procedure

- Disable the module:

```
# yum module disable module-name
```

  yum asks for confirmation and then disables the module with all its streams. All of the module streams become inactive. No installed content is removed.

APPENDIX A. YUM COMMANDS LIST

This chapter lists **YUM** commands for listing, installing, and removing content in Red Hat Enterprise Linux 9.

A.1. COMMANDS FOR LISTING CONTENT IN RHEL 9

The following table lists the commonly used **YUM** commands for finding content and its details in RHEL 9:

Command	Description
yum search *term*	Search for a package using term related to the package
yum repoquery *package*	Search available **YUM** repositories for a selected package
yum list --all	List information on all installed and available packages
yum list --installed **yum repoquery --installed**	List all packages installed on your system
yum list available **yum repoquery**	List all packages in all enabled repositories that are available to install
yum repolist	List all enabled repositories on your system
yum repolist --disabled	List all disabled repositories on your system
yum repolist --all	List both enabled and disabled repositories
yum repoinfo	List additional information about the repositories
yum info *package-name* **yum repoquery --info *package_name***	Display details of an available package
yum repoquery --info --installed *package_name*	Display details of a package installed on your system
yum module list	List available modules
yum module info *module-name*	Display details of a module

Command	Description
yum module info --profile *module-name*	List packages installed by profiles of a module using the default stream
yum module list *module-name*	Display the current status of a module
yum module provides *package*	Determine which modules provide a package. Note that if the package is available outside any modules, the output of this command is empty.
yum module info --profile *module-name:stream*	Display packages installed by profiles of a module using a specified stream
yum group summary	View the number of installed and available groups
yum group list	List all installed and available groups
yum group info *group-name*	List mandatory and optional packages contained in a particular group

A.2. COMMANDS FOR INSTALLING CONTENT IN RHEL 9

The following table lists the commonly used **YUM** commands for installing content in RHEL 9:

Command	Description
yum install *package-name*	Install a package. If the package is provided by a module stream, **yum** resolves the required module stream, and enables it automatically while installing this package. This also happens recursively for all package dependencies. If more module streams satisfy the requirement, the default ones are used.
yum install *package-name-1 package-name-2*	Install multiple packages and their dependencies simultaneously
yum install *package-name.arch*	Specify the architecture of the package by appending it to the package name when installing packages on a *multilib* system (AMD64, Intel 64 machine)
yum install */usr/sbin/binary-file*	Install a binary using the path to this binary as an argument

Command	Description
yum install ***/path/***	Install a previously-downloaded package from a local directory
yum module enable ***module-name***	Enable a module using its default stream. Enable the module when you want to make the packages available to the system but do not currently want to install any of them. Some modules might not define default streams. In such cases, you must explicitly specify the stream.
yum module enable ***module-name:stream***	Enable a module using a specific stream. If the module defines a default stream, you can omit the stream and colon.
yum module install ***module-name***	Install a module using the default stream and profiles Note that some modules do not define default streams.
yum module install ***module-name:stream***	Install a module using a specific stream and default profiles
yum module install ***module-name:stream/profile***	Install a module using a specific stream and profile
yum group install ***group-name***	Install a package group by a group name
yum group install ***groupID***	Install a package group by the groupID

A.3. COMMANDS FOR REMOVING CONTENT IN RHEL 9

The following table lists the commonly used **YUM** commands for removing content in RHEL 9:

Command	Description
yum remove ***package-name***	Remove a particular package and all dependent packages
yum remove ***package-name-1 package-name-2***	Remove multiple packages and their unused dependencies simultaneously
yum group remove group-name	Remove a package group by the group name
yum group remove ***groupID***	Remove a package group by the groupID

Command	Description
yum module remove --all *module-name:stream*	Remove all packages from an active stream
yum module remove *module-name:stream/profile*	Remove packages from an installed profile
yum module remove *module-name:stream*	Remove packages from all installed profiles within a stream
yum module reset *module-name*	Reset a module to the initial state
yum module disable *module-name*	Disable a module and all its streams