Red Hat Enterprise Linux 9

Managing software with the DNF tool

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

Last Updated: 2022-06-17

Red Hat Enterprise Linux 9 Managing software with the DNF tool

A guide to managing software with DNF 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 DNF tool in Red Hat Enterprise Linux 9. This includes a description of how to use modules, application streams, and profiles.

Table of Contents

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.

- 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 feedback via Bugzilla, create a new 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

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 **yum** instead of **dnf**, 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 9 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 9 installation
- Package manifest

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 with a shorter life cycle are listed in the Red Hat Enterprise Linux Application Streams Life Cycle page.

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 **dnf install** command.

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

It is recommended to review the Red Hat Enterprise Linux Application Stream Lifecycle definitions for any content lifecycle considerations.

NOTE

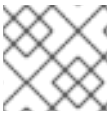
Not all modules are Application Streams. Dependencies of other modules are not considered Application Streams.

Additional resources

- Red Hat Enterprise Linux Life Cycle
- Red Hat Enterprise Linux Application Streams Life Cycle
- Package manifest

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.



NOTE

Modules will be available in future minor RHEL 9 releases.

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 can depend on other 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
- Life cycle length and your update plan

For per-component changes, see the Release Notes.

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 DNF

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

The following sections describe how to:

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

3.1. VIEWING THE CURRENT DNF CONFIGURATIONS

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

Procedure

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

```
# dnf config-manager --dump
```

3.2. SETTING DNF MAIN OPTIONS

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

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

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

3.3. USING DNF PLUG-INS

DNF 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 **DNF** plug-ins.

3.3.1. Managing DNF 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 **dnf** commands. If this option is missing, you can add it manually to the file.

Every installed plug-in may have 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 DNF plug-ins

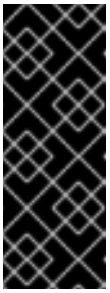
In the **DNF** tool, plug-ins are loaded by default.

The following procedure describes how to modify loading of **DNF** plug-ins, and enable or disable specific **DNF** plug-ins.

Procedure

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

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



IMPORTANT

Disabling all plug-ins is **not** advised. Certain plug-ins provide important **DNF** 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 **DNF**.

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

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

```
# dnf --noplugins update
```

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

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

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

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

```
# dnf 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.
- Section 4.7. "Specifying global expressions in dnf input" describes how to ensure global expressions are passed to **dnf** as intended.

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:

  ```
  # dnf search term
  ```

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

  Note that the **dnf search** command returns term matches within the name and summary of the packages. This makes the search faster and you can 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:

  ```
  # dnf search --all term
  ```

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

  Note that the **dnf 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 **dnf**.

Procedure

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

```
# dnf list --all
```

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

```
# dnf list --installed
```

Alternatively:

```
# dnf repoquery --installed
```

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

```
# dnf list --available
```

Alternatively:

```
# dnf repoquery
```

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

4.3. LISTING REPOSITORIES

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

Procedure

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

```
# dnf repolist
```

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

```
# dnf repolist --disabled
```

- To list both enabled and disabled repositories, use:

```
# dnf repolist --all
```

- To list additional information about the repositories, use:

```
# dnf 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 dnf input .

4.4. DISPLAYING PACKAGE INFORMATION

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

Procedure

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

  ```
  # dnf info package-name
  ```

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

  Alternatively:

  ```
  # dnf 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:

  ```
  # dnf 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 dnf input .

4.5. LISTING PACKAGE GROUPS

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

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

  ```
  # dnf group summary
  ```

- To list all installed and available groups, use:

  ```
  # dnf group list
  ```

  Note that you can filter the results by appending command line options for the **dnf 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:

  ```
  # dnf 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 dnf input .

4.6. LISTING AVAILABLE MODULES

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

Procedure

- To list module streams available to your system:

```
# dnf module list
```

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

NOTE

Modules will be available in future minor RHEL 9 releases.

Additional resources

- Modules
- Module streams
- Module profiles

4.7. SPECIFYING GLOBAL EXPRESSIONS IN DNF INPUT

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

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

Procedure

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

```
# dnf 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:

```
# dnf 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. "Running installed content" provides details for running RHEL 9 installed content.

5.1. INSTALLING PACKAGES

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

Procedure

- Install the package:

  ```
  # dnf install package-name
  ```

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

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

  ```
  # dnf 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:

  ```
  # dnf 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:

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

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

  dnf 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:

  ```
  # dnf install /path/
  ```

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

5.2. INSTALLING PACKAGE GROUPS

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

Procedure

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

  ```
  # dnf 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:

  ```
  # dnf group install groupID
  ```

  Replace *groupID* with the ID of the group.

5.3. 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
  ```

5.4. ADDITIONAL RESOURCES

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

CHAPTER 6. UPDATING RHEL 9 CONTENT

With **DNF** you can 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 **DNF** 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 **dnf**.

Procedure

- Run the following command to see which packages installed on your system have available updates:

```
# dnf 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 **dnf**.

Procedure

- To update all packages and their dependencies, use:

```
# dnf upgrade
```

- To update a single package, use:

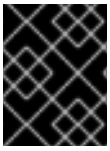
```
# dnf upgrade package-name
```

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

- To update a package group, use:

```
# dnf group upgrade group-name
```

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



IMPORTANT

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

6.3. UPDATING SECURITY-RELATED PACKAGES

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

Procedure

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

```
# dnf upgrade --security
```

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

```
# dnf upgrade-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 **DNF** 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 DNF AUTOMATIC

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

Procedure

- Install the **dnf-automatic** package:

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

Verification

- Verify the successful installation by confirming the presence of the **dnf-automatic** package:

```
# 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 **DNF**.

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 must always 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 **downloads**, **downloads and installs**, or **reports** available updates, use:

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

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

- 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

- Verify that the timer is enabled:

```
# 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
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** 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.

8.1. REMOVING INSTALLED PACKAGES

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

Procedure

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

  ```
  # dnf 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:

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

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

NOTE

dnf 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:

  ```
  # dnf group remove group-name
  ```

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

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

  ```
  # dnf group remove groupID
  ```

  Replace *groupID* with the ID of the group.

8.3. ADDITIONAL RESOURCES

- Commands for removing content in RHEL 9

CHAPTER 9. HANDLING PACKAGE MANAGEMENT HISTORY

With the **dnf history** command, you can review the following information

- Timeline of **DNF** 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 the transactions

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

The following section describes how to use **dnf** 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 **DNF** transactions, the latest operations for a selected package, and details of a particular transaction.

Procedure

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

```
# dnf history
```

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

```
# dnf 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 dnf input .

- To display details of a particular transaction, use:

```
# dnf 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 **dnf**.

Procedure

- To revert a particular transaction, use:

```
# dnf history undo transactionID
```

  Replace *transactionID* with the ID of the transaction.

- To revert the last transaction, use:

```
# dnf history undo last
```

Note that the **dnf history undo** command only reverts the steps that were performed during the transaction. If the transaction installed a new package, **dnf history undo** uninstalls it. If the transaction uninstalled a package, **dnf history undo** reinstalls it. The **dnf 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 **dnf**.

Procedure

- To repeat a particular transaction, use:

```
# dnf history redo transactionID
```

  Replace *transactionID* with the ID of the transaction.

- To repeat the last transaction, use:

```
# dnf history redo last
```

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

CHAPTER 10. MANAGING CUSTOM SOFTWARE REPOSITORIES

The configuration information for **DNF** and related utilities are stored in the **/etc/dnf/dnf.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/dnf/dnf.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 DNF repository options" describes how to set **[repository]** options.
- Section 10.2. "Adding a DNF repository" describes how to define a new **DNF** repository.
- Section 10.3. "Enabling a DNF repository" describes how to enable a **DNF** repository added to your system.
- Section 10.4. "Disabling a DNF repository" describes how to disable a **DNF** repository added to your system.

10.1. SETTING DNF REPOSITORY OPTIONS

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

NOTE

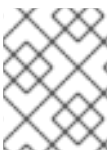
Do not give custom repositories names used by the Red Hat repositories to avoid conflicts.

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

10.2. ADDING A DNF REPOSITORY

To define a new repository, you can either:

- Add a **[*repository*]** section to the **/etc/dnf/dnf.conf** file.
- Add a **[*repository*]** section to a **.repo** file in the **/etc/yum.repos.d/** directory.
  Installed RPMs or software management tools, for example, Subscription Manager, can provide their own **.repo** file.



NOTE

Define your repositories in a **.repo** file instead of **/etc/dnf/dnf.conf** as all files with the **.repo** file extension in this directory are read by **dnf**.

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

Procedure

- Add a repository to your system:

```
# dnf 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 DNF REPOSITORY

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

Procedure

- Enable a repository:

```
# dnf config-manager --enable repositoryID
```

Replace *repositoryID* with the unique repository ID.

Additional resources

Listing repositories

10.4. DISABLING A DNF REPOSITORY

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

Procedure

- Disable a repository:

```
# dnf config-manager --disable repositoryID
```

Replace *repositoryID* with the unique repository ID.

Additional resources

Listing repositories

APPENDIX A. DNF COMMANDS LIST

This chapter lists **DNF** 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 **DNF** commands for finding content and its details in RHEL 9:

Command	Description
dnf search *term*	Search for a package using term related to the package
dnf repoquery *package*	Search available **DNF** repositories for a selected package
dnf list	List information on all installed and available packages
dnf list --installed **dnf repoquery --installed**	List all packages installed on your system
dnf list --available **dnf repoquery**	List all packages in all enabled repositories that are available to install
dnf repolist	List all enabled repositories on your system
dnf repolist --disabled	List all disabled repositories on your system
dnf repolist --all	List both enabled and disabled repositories
dnf repoinfo	List additional information about the repositories
dnf info *package-name* **dnf repoquery --info *package_name***	Display details of an available package
dnf repoquery --info --installed *package_name*	Display details of a package installed on your system
dnf module list	List modules and their current status Note that if the package is available outside any modules, the output of this command is empty.
dnf group summary	View the number of installed and available groups

Command	Description
dnf group list	List all installed and available groups
dnf group info *group-name*	List mandatory and optional packages included in a particular group

A.2. COMMANDS FOR INSTALLING CONTENT IN RHEL 9

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

Command	Description
dnf install *package-name*	Install a package. If the package is provided by a module stream, **dnf** 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.
dnf install *package-name-1 package-name-2*	Install multiple packages and their dependencies simultaneously
dnf 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)
dnf install */usr/sbin/binary-file*	Install a binary using the path to the binary as an argument
dnf install */path/*	Install a previously downloaded package from a local directory

Command	Description
dnf install *package-url*	Install a remote package using a package URL 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. If the module defines a default stream, you can omit the stream and colon. Note that running this command does not install any RPM packages. Note that some modules do not define default streams. Note that running this command also enables the specified stream.
dnf group install *group-name*	Install a package group by a group name
dnf 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 **DNF** commands for removing content in RHEL 9:

Command	Description
dnf remove *package-name*	Remove a particular package and all dependent packages
dnf remove *package-name-1 package-name-2*	Remove multiple packages and their unused dependencies simultaneously
dnf group remove group-name	Remove a package group by the group name
dnf group remove *groupID*	Remove a package group by the groupID Note that running this command can remove critical packages from your system. Note that running this command does not remove packages from the specified module. Note that running this command does not remove packages from the specified module.