# Red Hat Enterprise Linux 8

## Installing, managing, and removing user space components

An introduction to AppStream and BaseOS in Red Hat Enterprise Linux 8

Last Updated: 2019-05-09

# Red Hat Enterprise Linux 8 Installing, managing, and removing user space components

An introduction to AppStream and BaseOS in Red Hat Enterprise Linux 8

## Legal Notice



## Abstract

This document describes searching, discovering, installing, and using content in the AppStream and BaseOS repositories in Red Hat Enterprise Linux 8. This includes a description of how to use modules, application streams, and profiles.

# Table of Contents

# 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, make sure you are viewing the documentation in the Multi-page HTML format. Highlight the part of text that you want to comment on. Then, click the **Add Feedback** pop-up that appears below the highlighted text, and 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. USING APPSTREAM

The following sections provide an overview of the concepts related to the AppStream repository in Red Hat Enterprise Linux 8.

- Section 1.1, "Distribution of content in RHEL 8" describes how content in Red Hat Enterprise Linux 8 is split into BaseOS and AppStream.
- Section 1.2, "Application Streams" describes the concept of Application Streams.
- Section 1.3, "Packaging methods in RHEL 8" describes the types of content provided by AppStream.
- Section 1.4, "Package management using YUM in RHEL 8" describes how the **YUM** package manager provided in Red Hat Enterprise Linux 8 combines the traditional and modular features.

## 1.1. DISTRIBUTION OF CONTENT IN RHEL 8

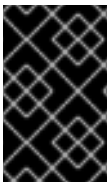
RHEL 8 content is distributed through the two main repositories: **BaseOS** and **AppStream**.

**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. Content in AppStream is available in one of two formats - the familiar RPM format and an extension to the RPM format called *modules*.



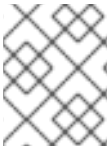
**IMPORTANT**

Both BaseOS and AppStream content sets are required for a basic RHEL installation, and are available with all RHEL subscriptions. For installation instructions, see the Installing and Deploying RHEL document.

## 1.2. APPLICATION STREAMS

Red Hat Enterprise Linux 8 introduces the concept of Application Streams - versions of user space components. Multiple versions of these components are now delivered and updated more frequently than the core operating system packages. This provides greater flexibility to customize Red Hat Enterprise Linux without impacting the underlying stability of the platform or specific deployments.

Components made available as Application Streams can be packaged as modules or RPM packages and are delivered through the AppStream repository in Red Hat Enterprise Linux 8. Each ApplStream component has a given life cycle.



**NOTE**

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

Each module defines its own lifecycle which is closer to the natural life of the application rather than the RHEL lifecycle. To find out what Application Streams are available and what is their length of support, see the Red Hat Enterprise Linux 8 Application Streams Life Cycle page.

**Additional Resources**

- Red Hat Enterprise Linux 8 Application Streams Life Cycle
- Red Hat Enterprise Linux Life Cycle

## 1.3. PACKAGING METHODS IN RHEL 8

The AppStream repository contains content packaged in two ways:

**Individual RPM packages**

Traditional RPM packages available for immediate installation.

**Modules**

Modules are collections of packages representing a logical unit: an application, a language stack, a database, or a set of tools. These packages are built, tested, and released together.

## 1.4. PACKAGE MANAGEMENT USING YUM IN RHEL 8

The **YUM** package management tool is now based on the DNF technology and it adds support for the new modular features.

Usage of **YUM** has not been changed when handling individual RPM packages. For handling the modular content, the **yum module** command has been added. See Chapter 4, *Installing RHEL 8 content* for additional details.

Where required, the modular functionality automatically selects the appropriate combination of modules and streams to enable installation of logical sets of packages for convenient usage.

# CHAPTER 2. INTRODUCTION TO MODULES

Besides individual RPM packages, the AppStream repository contains 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.

The subsequent sections describe further features for organization and handling of content within modules:

- Streams - organization of content by version. For more details see Section 2.1, “Module streams”.
- Profiles - organization of content by purpose. For more details see Section 2.2, “Module profiles”.

## 2.1. 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 either if marked as default or 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 may have a default stream which usually provides the latest or recommended version of the component. 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.

> **Example 2.1. postgresql module streams**
>
> The **postgresql** module provides the **PostgreSQL** database versions 9.6 and 10 in the respective streams **9.6** and **10**. Stream **10** is currently the default one. This means that the system will attempt to install the **postgresql-10.6** package if asked for **postgresql**.

**Additional resources**

- For more information about modular dependencies, see Section 6.1, “Modular dependencies and stream changes”.

## 2.2. 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, one of its profiles can be marked as *default* and is then used for profile installation actions when no other profile is explicitly specified. However, existence of a default profile for a module stream is not required.

**Example 2.2. httpd module profiles**

The **httpd** module, which provides the **Apache** web server, offers the following profiles for installation:

- **common** - a hardened production-ready deployment, the default profile
- **devel** - the packages necessary for making modifications to **httpd**
- **minimal** - the smallest set of packages that will provide a running webserver

# CHAPTER 3. FINDING RHEL 8 CONTENT

The following sections describe how to locate and examine content in the AppStream and BaseOS repositories in Red Hat Enterprise Linux 8.

- Section 3.1, "Searching for a package" describes how to search for packages providing desired content.
- Section 3.2, "Listing available modules" describes how to list available modules and find out details about them.
- Section 3.3, "Example: Finding out details about a module" contains an example of steps needed to examine a module in more detail.
- Section 3.4, "Commands for listing content" provides a reference of the commands useful for inspecting content.

## 3.1. SEARCHING FOR A PACKAGE

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

**Prerequisites**

- Name of the desired application or content must be known

**Procedure**

1. Search for a package with a text string, such as application name:

```
$ yum search "text string"
```

2. View details about a package:

```
$ yum info package
```

## 3.2. LISTING AVAILABLE MODULES

This section describes steps needed for finding what modules are available and what their details are.

**Procedure**

1. 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.

2. 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
```

3. Optional: You can also list which of these packages are installed by each of module profiles:

```
$ yum module info --profile module-name
```

4. Display the current status of a module, including enabled streams and installed profiles:

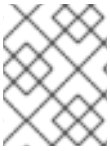
```
$ yum module list module-name
```

**Additional resources**

- Chapter 2, *Introduction to modules*

## 3.3. EXAMPLE: FINDING OUT DETAILS ABOUT A MODULE

This example shows how to locate a module in the AppStream repository and how to find out more about its contents.



**NOTE**

The outputs in this example have been edited for brevity. Actual outputs may contain more information than shown here.

**Procedure**

1. List available modules:

```
$ yum module list
Name       Stream  Profiles    Summary
(...)
postgresql  10 [d]  client,     PostgreSQL server and client module
                    server [d]
postgresql  9.6     client,     PostgreSQL server and client module
                    server [d]
(...)

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

2. Examine details of the **postgresql** module:

```
$ yum module info postgresql

Name            : postgresql
Stream          : 10 [d][a]
Version         : 820190104140132
Context         : 9edba152
Profiles        : client, server [d]
Default profiles : server
Repo            : appstream-internal-nightly
Summary         : PostgreSQL server and client module
Description     : (...)
(...)

Name            : postgresql
```

```
Stream          : 9.6
Version         : 820190104140337
Context         : 9edba152
Profiles        : client, server [d]
Default profiles : server
Repo            : appstream-internal-nightly
Summary         : PostgreSQL server and client module
Description     : (...)
(...)

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled, [a]ctive]
```

Because no stream is specified, all streams are used for the listing.

3. Examine profiles available in stream **10** of the **postgresql** module:

```
$ yum module info --profile postgresql:10
(...)
Name   : postgresql:10:820190104140132:9edba152:x86_64
client : postgresql
server : postgresql-server
```

Each of the profiles installs a different set of packages, including their dependencies.

4. Install the **postgresql** module using the default stream **10** and profile **server**:

```
# yum install @postgresql
Dependencies resolved.
================================================================================
 Package          Version                      Repository Size
================================================================================
Installing group/module packages:
 postgresql-server 10.6-1.module+el8+2469+5ecd5aae appstream  5.1 M
Installing dependencies:
 libpq            10.5-1.el8                   appstream  188 k
 postgresql       10.6-1.module+el8+2469+5ecd5aae appstream  1.5 M
Installing module profiles:
 postgresql/server
Enabling module streams:
 postgresql               10

Transaction Summary
================================================================================
Install  3 Packages

Total download size: 6.7 M
Installed size: 27 M
Is this ok [y/N]: y
(...)
```

The stream **10** is enabled and packages in its profile **server** installed.

5. Inspect the current status of the **postgresql** module:

```
$ yum module list postgresql
Name       Stream    Profiles                Summary
```

```
postgresql 10 [d][e] client, server [d] [i]  (...)
postgresql 9.6       client, server [d]      (...)

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

The output shows that the default stream **10** is enabled and its profile **server** is installed.

## 3.4. COMMANDS FOR LISTING CONTENT

This section lists commonly used commands for finding content and its details.

**Command list**

**List available packages**

```
$ yum list available
```

**Search for a package using arbitrary text string**

```
$ yum search "text string"
```

**Display details for a package**

```
$ yum info package
```

**Find out which modules provide a package**

```
$ yum module provides package
```

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

**List available modules**

```
$ yum module list
```

**Display details of a module**

```
$ yum module info module-name
```

**List packages installed by profiles of a module using the default stream**

```
$ yum module info --profile module-name
```

**Display packages installed by profiles of a module using a specified stream**

```
$ yum module info --profile module-name:stream
```

**Display the current status of a module**

```
$ yum module list module-name
```

# CHAPTER 4. INSTALLING RHEL 8 CONTENT

The following sections describe how to install content in Red Hat Enterprise Linux 8.

- Section 4.1, “Installing a package” includes steps for installing a package.
- Section 4.2, “Selecting a stream before installation of packages” describes how to select a stream for package installation.
- Section 4.3, “Installing a module stream” describes steps to install sets of packages provided by modules.
- Section 4.4, “Example: Installing a non-default stream of an application” shows an example of steps needed to install a set of packages in a non-default version.
- Section 4.5, “Running installed content” provides details for running RHEL 8 installed content.
- Section 4.6, “Commands for installing RHEL 8 content” provides a reference of commands useful for installing RHEL 8 content.

## 4.1. INSTALLING A PACKAGE

This section describes how to install packages.

**Procedure**

- Install the package:

```
# yum install 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 an 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.

**Additional resources**

- Section 4.3, “Installing a module stream”
- Section 1.4, “Package management using YUM in RHEL 8”

## 4.2. SELECTING A STREAM BEFORE INSTALLATION OF PACKAGES

Default module streams ensure that users can install packages without caring about the modular features. When the user wants packages with version from a non-default stream, that stream must be enabled before packages provided by it can be installed.

**Prerequisites**

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

**Procedure**

- Enable the 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 (default or enabled), it is no longer active.

## 4.3. INSTALLING A MODULE STREAM

This section describes using a module stream to install the recommended set of packages from that module.

**Prerequisites**

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

**Procedure**

- Install a profile of the module stream:

```
# yum 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.

  Omit */profile* to use the default profile. If no profile is set as default, this step fails without a specified profile and you must specify it.

  Additionally, omit :*stream* to use the active stream. If there is no active stream for the module, you must specify a stream.

**Additional resources**

- Chapter 2, *Introduction to modules*
- Section 4.6, "Commands for installing RHEL 8 content"

## 4.4. EXAMPLE: INSTALLING A NON-DEFAULT STREAM OF AN APPLICATION

This example shows how to install an application from a non-default stream (version).

More specifically, this example shows how to install the **PostgreSQL** server (package **postgresql-server**) in version **9.6**, while the default stream provides version **10**.

**Procedure**

1. List modules that provide the **postgresql-server** package to see what streams are available:

```
$ yum module list postgresql
Name        Stream  Profiles              Summary
postgresql  10 [d]  client, default [d]  PostgreSQL server and client module
postgresql  9.6     client, default [d]  PostgreSQL server and client module

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

   The output shows that the **postgresql** module is available with streams **10** and **9.6**. The default stream is **10**.

2. Install the packages provided by the **postgresql** module in stream **9.6**:

```
# yum install @postgresql:9.6
Dependencies resolved.
================================================================================
 Package           Version                        Repository Size
================================================================================
Installing group/module packages:
 postgresql-server 9.6.10-1.module+el8+2470+d1bafa0e appstream  5.0 M
Installing dependencies:
 libpq             10.5-1.el8                     appstream  188 k
 postgresql        9.6.10-1.module+el8+2470+d1bafa0e appstream  1.4 M
Installing module profiles:
 postgresql/server
Enabling module streams:
 postgresql                9.6

Transaction Summary
================================================================================
Install  3 Packages

Total download size: 6.6 M
Installed size: 27 M
Is this ok [y/N]: y
(...)
Complete!
```

   Because the installation profile was not specified, the default profile **server** was used.

3. Verify the installed version of **PostgreSQL**:

```
$ postgres --version
postgres (PostgreSQL) 9.6.10
```

## 4.5. RUNNING INSTALLED CONTENT

Usually, after you install content from RHEL 8 repositories, new commands will be enabled. If the commands originated from a RPM package or RPM packages enabled by a module the experience of using the command should be no different. To run the new commands use them directly:

```
$ command
```

## 4.6. COMMANDS FOR INSTALLING RHEL 8 CONTENT

This section lists commonly used commands for installing RHEL 8 content.

**Command list**

**Install a package**

```
# yum install 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 happens recursively for all package dependencies, too. If more module streams satisfy the requirement, the default ones are used.

**Enable a module using its default stream**

```
# yum module enable module-name
```

Enable the module when you wish to make the packages available to the system but do not, at this time, wish to install any of them.

Some modules may not define default streams. In such case, you must explicitly specify the stream.

**Enable a module using a specific stream**

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

If the module defines a default stream, you can omit the stream and colon.

**Install a module using the default stream and profiles**

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

Alternatively:

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

CAUTION

Some modules do not define default streams.

**Install a module using a specific stream and default profiles**

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

Alternatively:

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

**Install a module using a specific stream and profile**

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

Alternatively:

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

## 4.7. ADDITIONAL RESOURCES

**Online resources**

- For more information about the traditional software installation methods, see Installing software with yum in Configuring basic system settings.

**Installed resources**

- For details of various **yum** tool commands, see the **yum(8)** manual page:

```
$ man yum
```

# CHAPTER 5. REMOVING RHEL 8 CONTENT

The following sections describe how to remove content in Red Hat Enterprise Linux 8:

- Section 5.1, "Removing installed packages" describes removing a package.
- Section 5.2, "Removing installed modules" describes removing content installed from a module stream.
- Section 5.3, "Resetting module streams" describes resetting module streams to initial state.
- Section 5.4, "Commands for removing content" summarizes the commands for removing content.

## 5.1. REMOVING INSTALLED PACKAGES

This section describes how to remove packages.

**Procedure**

- Remove the package:

```
# yum remove package
```

The package is removed together with any other dependent packages.

## 5.2. REMOVING INSTALLED MODULES

Removing a module removes all of the packages installed by profiles of the currently enabled module stream, and any further packages and modules that depend on these.

Packages installed from this module stream not listed in any of its profiles remain installed on the system and can be removed manually.

**Prerequisites**

- A module which you want removed must have already installed some profiles.
- You must understand modular dependency resolution.

**Procedure**

1. List packages installed from the module:

```
$ yum module info module-name | grep module+el8 | sed 's/.*: //g;s/\n/ /g' | xargs yum list installed
```

Replace *module-name* with the name of the module. This will list all packages installed from this module.

2. Remove the packages listed in the previous step:

```
# yum remove package
```

Replace *package* with the packages listed in previous step. You can supply multiple package names separated by spaces. The **yum** tool will present a summary of the changes and ask for confirmation.

3. Mark the module profiles as uninstalled:

```
# yum module remove module-name
```

The currently enabled module stream remains enabled.

4. Optionally, reset or disable the stream.

## 5.3. 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.

**Procedure**

- Reset the module state:

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

All streams of the module are returned to the initial state. No installed content is removed.

## 5.4. COMMANDS FOR REMOVING CONTENT

This section lists commonly used commands for removing content.

**Command list**

**Remove a package**

```
# yum remove package
```

**Remove installed module stream profiles**

```
# yum module remove module-name
```

**Reset all streams of a module to initial state**

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

**Disable a module and all its streams**

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

# CHAPTER 6. 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 6.1, "Modular dependencies and stream changes" describes modular dependency rules.
- Section 6.2, "Interaction of modular and non-modular dependencies" provides details for how the dependencies of module streams affect handling of package dependencies.
- Section 6.3, "Resetting module streams" provides steps for resetting modules to their initial state.
- Section 6.4, "Disabling all streams of a module" provides steps for disabling completely a module and all its streams.
- Section 6.5, "Switching module streams to install a different version of content" describes how to install a different version of content by switching active streams for a module when some content is already installed.

## 6.1. 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 the packages declare. Additionally, all module stream dependencies must be satisfied.

As a result:

- Enabling a module stream can require enabling streams of further modules.
- Installing a module stream profile or installing packages from a stream can require enabling streams of further modules and installing further packages.
- Disabling a stream of a module 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.

It is not possible to enable a stream of a module when another stream of the same module is already enabled. To switch streams, you must first reset the module, and then enable the new stream. Removing all packages installed from a stream before switching to a different stream is highly recommended, because it prevents the system from reaching states where packages could be installed with no repository or stream providing them.

Technically, resetting module does not automatically change any installed packages. Removing the packages provided by the previous stream and any packages that depend on them is an explicit manual operation.

## 6.2. 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 not only resolution of the RPM dependencies, but also the modular dependencies must be resolved beforehand.

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.

## 6.3. 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.

**Procedure**

- Reset the module state:

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

  All streams of the module are returned to the initial state. No installed content is removed.

## 6.4. DISABLING ALL STREAMS OF A MODULE

Modules which 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.

**Prerequisites**

- You must 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.

## 6.5. SWITCHING MODULE STREAMS TO INSTALL A DIFFERENT VERSION OF CONTENT

Switching to a different module stream usually means upgrading or downgrading the packages to a different version than the current version installed on your system.

The general intent of this procedure is to:

- first remove the already installed content,
- then change the enabled streams,
- and finally install the new content.

This sequence reduces the amount of actions where dependencies could prevent you from continuing and instead concentrates them into a few steps.

**Prerequisites**

- A module stream must be active, and another stream of the module must exist.
- You must understand modular dependency resolution.
- Keep notes of the changes you are making so that you can undo them.
- Remember that package and modular dependencies might prevent you from completing this procedure, just as package dependencies could do in previous versions of Red Hat Enterprise Linux.

**Procedure**

1. Remove the content installed from the currently active module stream and its dependencies:
   a. List the packages installed from the module:

   ```
   $ yum module info module-name | grep module+el8 | sed 's/.*: //g;s/\n/ /g' | xargs yum list installed
   ```

   This will list all packages installed from this module.

   b. Remove the packages listed in the previous step:

   ```
   # yum remove package
   ```

   Replace *package* with the packages listed in previous step. You can supply multiple package names separated by spaces.

   Package dependencies may force you to remove the dependent packages, too. Make sure to note these packages, find which modules provided them using the **yum module provides** command, and keep the list for later steps.

   c. Mark the module profiles as uninstalled:

   ```
   # yum module remove module-name
   ```

2. Reset the module and its streams:

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

3. Enable the new stream and resolve modular dependencies:

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

The **yum** tool will present a summary of the changes required by package and modular dependencies and ask for confirmation. It is possible that modular dependencies can prevent you from taking this action immediately. In that case, apply this procedure recursively to further module streams, or reset them. To find these streams, refer to the output of the command for this step, and to the list you made in step 1.

To continue in this procedure, you must succeed in enabling the new stream. If the combined dependencies prevent you from doing so, revert your changes: Enable the original streams and install back the packages.

4. Install content from the new stream of the module.

   a. Install profiles from the new stream:

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

   b. Install individual packages provided by the new stream:

```
# yum module install package
```