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

**MAKING OPEN SOURCE MORE INCLUSIVE** .................................................. 3

**PROVIDING FEEDBACK ON RED HAT DOCUMENTATION** .......................... 4

**CHAPTER 1. USING APPSTREAM** ....................................................... 5
  1.1. DISTRIBUTION OF CONTENT IN RHEL 8 ...................................... 5
  1.2. APPLICATION STREAMS ............................................................ 5
  1.3. PACKAGING METHODS IN RHEL 8 ............................................. 6
  1.4. PACKAGE MANAGEMENT USING YUM IN RHEL 8 ......................... 6

**CHAPTER 2. INTRODUCTION TO MODULES** .......................................... 7
  2.1. MODULE STREAMS ................................................................. 7
  2.2. MODULE PROFILES .................................................................... 8

**CHAPTER 3. FINDING RHEL 8 CONTENT** ............................................. 9
  3.1. SEARCHING FOR A PACKAGE .................................................... 9
  3.2. LISTING AVAILABLE MODULES ............................................... 9
  3.3. COMMANDS FOR LISTING CONTENT ......................................... 12

**CHAPTER 4. INSTALLING RHEL 8 CONTENT** ........................................ 14
  4.1. INSTALLING A PACKAGE .......................................................... 14
  4.2. SELECTING A STREAM BEFORE INSTALLATION OF PACKAGES ......... 14
  4.3. OVERRIDING MODULE DEFAULT STREAMS ................................ 15
  4.4. INSTALLING MODULAR CONTENT ............................................. 16
  4.5. RUNNING INSTALLED CONTENT ................................................ 18
  4.6. COMMANDS FOR INSTALLING RHEL 8 CONTENT ......................... 18
  4.7. ADDITIONAL RESOURCES ........................................................... 19

**CHAPTER 5. REMOVING RHEL 8 CONTENT** .......................................... 20
  5.1. REMOVING INSTALLED PACKAGES .......................................... 20
  5.2. REMOVING INSTALLED MODULAR CONTENT ................................ 20
  5.2.1. Removing all packages from a module stream ....................... 20
  5.2.2. Removing packages from an installed profile ......................... 23
  5.3. resetting module streams ......................................................... 25
  5.4. COMMANDS FOR REMOVING CONTENT .................................... 26

**CHAPTER 6. MANAGING VERSIONS OF APPLICATION STREAM CONTENT** .... 27
  6.1. MODULAR DEPENDENCIES AND STREAM CHANGES ...................... 27
  6.2. INTERACTION OF MODULAR AND NON-MODULAR DEPENDENCIES .... 28
  6.3. RESETING MODULE STREAMS .................................................... 28
  6.4. DISABLING ALL STREAMS OF A MODULE .................................... 28
  6.5. SWITCHING TO A LATER STREAM ............................................... 28
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. 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 form of RPM packages 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 packages 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 Performing a standard RHEL installation 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 Application Stream has a given life cycle, either the same as RHEL 8 or shorter, more suitable to the particular application. Application Streams with a shorter life cycle are listed in the Red Hat Enterprise Linux 8 Application Streams Life Cycle page.
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 8 Application Streams 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 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.

**IMPORTANT**

The default stream is designed to be unchanging throughout the RHEL major release. Make sure to consider each stream’s life cycle. Do not rely on the default stream for instances in which the default stream reaches the End of Life status prior to the end of the RHEL major release.

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 it
- Compatibility
- Life cycle length and your update plan

For a list of all available modules and streams, see the Package manifest. For per-component changes, see the Release Notes.
Example 2.1. `postgresql` module streams

The `postgresql` module provides the PostgreSQL database versions 9.6, 10, and 12 in the respective streams 9.6, 10, and 12. 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".
- For instructions on how to upgrade module streams, see Section 6.5, "Switching to a later stream".

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, 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.

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.
- Example 3.1, “Finding out details about a module” contains an example of steps needed to examine a module in more detail.
- Section 3.3, “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

- Introduction to modules

Example 3.1. 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 9.6 client, server [d] PostgreSQL server and client module
   postgresql 10 [d] client, server [d] PostgreSQL server and client module
   postgresql 12 client, server [d] PostgreSQL server and client module
   (....)

   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
   Summary : PostgreSQL server and client module
   Description : (....)

   (....)
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 module install postgresql
```

```
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 9.6       client, server [d]      (...)
postgresql 10 [d][e] client, server [d] [i]  (...)
postgresql 12        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.

3.3. COMMANDS FOR LISTING CONTENT

Following are the commonly used commands for finding content and its details.

**List available packages**

```
$ yum list available
```

**Search available YUM repositories for a selected package**

```
$ yum repoquery package
```

**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, “Overriding module default streams”** describes how to override repository module defaults on a per-system basis.
- **Section 4.4, “Installing modular content”** describes steps to install sets of packages provided by modules, streams, and profiles.
- **Example 4.3, “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**

- Installing modular content
- 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:

```bash
# 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.

### 4.3. OVERRIDING MODULE DEFAULT STREAMS

By default, the `yum` utility uses the module default streams defined in the repository which contains the modules. It is possible to override 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.

```yaml
---
document: modulemd-defaults
version: 1
data:
  module: postgresql
  stream: "10"
  profiles:
    10: [server]
    12: [server]
    13: [server]
    9.6: [server]
    ...
```

The output above represents the default definition present for the `postgresql` module at the time of this writing.

**Example 4.1. Example postgresql module with original defaults**

```bash
# yum module list postgresql
(…)
Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMs)
Name        Stream  Profiles      Summary
```
2. To set the default stream to 13, the following configuration can be implemented:

```yaml
---
m: modulemd-defaults
v: 1
data:
  module: "postgresql"
  stream: "13"
profiles:
  10: [server]
  12: [server]
  13: [server]
  9.6: [server]
---
```

Example 4.2. Example `postgresql` with module default overridden

```
# yum module list postgresql
(...)
Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMs)
Name   Stream   Profiles                Summary
postgresql  9.6 client, server [d]  PostgreSQL server and client module
postgresql 10 [d] client, server [d]  PostgreSQL server and client module
postgresql 12 client, server [d]  PostgreSQL server and client module
postgresql 13 [d] client, server [d]  PostgreSQL server and client module
```

4.4. INSTALLING MODULAR CONTENT

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

**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 an active module stream (the default one or 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

- Introduction to modules
- Commands for installing RHEL 8 content

Example 4.3. 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 9.6  client, server [d] PostgreSQL server and client module
   postgresql 10 [d] client, server [d] PostgreSQL server and client module
   postgresql 12  client, server [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 9.6, 10 and 12. The default stream is 10.

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

   ```
   # yum module 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
   ```
Because the installation profile was not specified, the default profile `server` was used.

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

```bash
$ 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:

```bash
$ 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**

```bash
# 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**

```bash
# 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**

```bash
# 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**

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

Alternatively:

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

**IMPORTANT**

Some modules do not define default streams.

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

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

Alternatively:

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

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

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

Alternatively:

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

### 4.7. ADDITIONAL RESOURCES

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

- For details of various *yum* tool commands, see the *yum(8)* manual page.
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 modular content”** describes removing content installed from a module stream or a profile.
- **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:
  
```bash
  # yum remove package
  ```

  The package is removed together with any other dependent packages.

5.2. REMOVING INSTALLED MODULAR CONTENT

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

5.2.1. 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, including 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 must understand modular dependency resolution.

**Procedure**
1. Remove all packages from a selected stream:

```bash
# 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 5.2.2**, “Removing packages from an installed profile”.

**Example 5.1. Removing packages from the whole stream**

This example shows how to remove all packages from the module stream.

**Procedure**

1. Install the **php:7.3** module stream, including all available profiles:

```bash
[root@rhel-8 ~]# yum module install php:7.3/*
```

Updating Subscription Management repositories.
Dependencies resolved.

```
Package Arch Version Repository Size
```

```
Installing group/module packages:
libzip x86_64 1.5.2-1.module+el8.1.0+3189+a1bff096 rhel-8-for-x86_64-appstream-rpms 63 k
php-cli x86_64 7.3.5-5.module+el8.1.0+4560+e0eee7d6 rhel-8-for-x86_64-appstream-rpms 3.0 M
php-common x86_64 7.3.5-5.module+el8.1.0+4560+e0eee7d6 rhel-8-for-x86_64-appstream-rpms 663 k
php-devel x86_64 7.3.5-5.module+el8.1.0+4560+e0eee7d6 rhel-8-for-x86_64-appstream-rpms 735 k
php-fpm x86_64 7.3.5-5.module+el8.1.0+4560+e0eee7d6 rhel-8-for-x86_64-appstream-rpms 1.6 M
php-json x86_64 7.3.5-5.module+el8.1.0+4560+e0eee7d6 rhel-8-for-x86_64-appstream-rpms 73 k
php-mbstring x86_64 7.3.5-5.module+el8.1.0+4560+e0eee7d6 rhel-8-for-x86_64-appstream-rpms 610 k
php-pear noarch 1:1.10.9-1.module+el8.1.0+3189+a1bff096 rhel-8-for-x86_64-appstream-rpms 359 k
php-pecl-zip x86_64 1.15.4-1.module+el8.1.0+3189+a1bff096 rhel-8-for-x86_64-appstream-rpms 51 k
php-process x86_64 7.3.5-5.module+el8.1.0+4560+e0eee7d6 rhel-8-for-x86_64-appstream-rpms 84 k
php-xml x86_64 7.3.5-5.module+el8.1.0+4560+e0eee7d6 rhel-8-for-x86_64-appstream-rpms 188 k
Installing dependencies:
autoconf noarch 2.69-27.el8 rhel-8-for-x86_64-appstream-rpms 710
```
Installing weak dependencies:
perl-IO-Socket-IP
  noarch 0.39-5.el8          rhel-8-for-x86_64-appstream-rpms 47 k

Installing module profiles:
  php/common
  php/devel
  php/minimal

Enabling module streams:
  httpd 2.4
  nginx 1.14
  php 7.3

Transaction Summary
========================================================================
  =
  Install  73 Packages

  Total download size: 76 M
  Installed size: 220 M
  Is this ok [y/N]: y

2. Remove all packages from the php:7.3 module stream:

[root@rhel-8 ~]# yum module remove php:7.3 --all

Updating Subscription Management repositories.
Dependencies resolved.
========================================================================
  =
  Package    Arch    Version                                      Repository Size
  ================================================================================
  =
  Removing:
  libzip      x86_64  1.5.2-1.module+el8.1.0+3189+a1bff096             @rhel-8-for-x86_64-appstream-rpms 313 k
  php-cli     x86_64  7.3.5-5.module+el8.1.0+4560+e0ee7d6              @rhel-8-for-x86_64-appstream-rpms 11 M
  php-common  x86_64  7.3.5-5.module+el8.1.0+4560+e0ee7d6              @rhel-8-for-x86_64-appstream-rpms 6.5 M
  php-devel   x86_64  7.3.5-5.module+el8.1.0+4560+e0ee7d6              @rhel-8-for-x86_64-appstream-rpms 5.3 M
  php-fpm     x86_64  7.3.5-5.module+el8.1.0+4560+e0ee7d6              @rhel-8-for-x86_64-appstream-rpms 5.6 M
  php-json    x86_64  7.3.5-5.module+el8.1.0+4560+e0ee7d6              @rhel-8-for-x86_64-appstream-rpms 53 k
  php-mbstring x86_64  7.3.5-5.module+el8.1.0+4560+e0ee7d6              @rhel-8-for-x86_64-appstream-rpms 1.9 M
  php-pear    noarch  1:1.10.9-1.module+el8.1.0+3189+a1bff096          @rhel-8-for-x86_64-appstream-rpms 2.1 M
  php-pecl-zip x86_64  1.15.4-1.module+el8.1.0+3189+a1bff096           @rhel-8-for-x86_64-appstream-rpms 119 k
  php-process x86_64  7.3.5-5.module+el8.1.0+4560+e0ee7d6              @rhel-8-for-x86_64-appstream-rpms 117 k
5.2.2. 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, including 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 must understand modular dependency resolution.

Procedure

1. Uninstall packages belonging to the selected profile:

   ```bash
   # 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:

   ```bash
   # 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 5.2.1, “Removing all packages from a module stream”.

Example 5.2. Removing packages from a selected profile

This example shows how to remove packages belonging only to a selected profile.

**Procedure**

1. Install the **php:7.3** module stream, including all available profiles:

   ```bash
   [root@rhel-8 ~]# yum module install php:7.3/
   Installing Subscription Management repositories.
   Dependencies resolved.
   =
   Package          Arch   Version                              Repository                        Size
   =
   Installing group/module packages:
   libzip           x86_64 1.5.2-1.module+el8.1.0+3189+a1bff096 rhel-8-for-x86_64-appstream-rpms 63 k
   php-cli          x86_64 7.3.5-5.module+el8.1.0+4560+e0eee7d6 rhel-8-for-x86_64-appstream-rpms 3.0 M
   php-common       x86_64 7.3.5-5.module+el8.1.0+4560+e0eee7d6 rhel-8-for-x86_64-appstream-rpms 663 k
   php-devel        x86_64 7.3.5-5.module+el8.1.0+4560+e0eee7d6 rhel-8-for-x86_64-appstream-rpms 735 k
   php-fpm           x86_64 7.3.5-5.module+el8.1.0+4560+e0eee7d6 rhel-8-for-x86_64-appstream-rpms 1.6 M
   php-json         x86_64 7.3.5-5.module+el8.1.0+4560+e0eee7d6 rhel-8-for-x86_64-appstream-rpms 73 k
   php-mbstring     x86_64 7.3.5-5.module+el8.1.0+4560+e0eee7d6 rhel-8-for-x86_64-appstream-rpms 610 k
   php-pear         noarch 1:1.10.9-1.module+el8.1.0+3189+a1bff096
                     rhel-8-for-x86_64-appstream-rpms 359 k
   php-pecl-zip     x86_64 1.15.4-1.module+el8.1.0+3189+a1bff096
                     rhel-8-for-x86_64-appstream-rpms 51 k
   php-process      x86_64 7.3.5-5.module+el8.1.0+4560+e0eee7d6 rhel-8-for-x86_64-appstream-rpms 84 k
   php-xml          x86_64 7.3.5-5.module+el8.1.0+4560+e0eee7d6 rhel-8-for-x86_64-appstream-rpms 188 k
   Installing dependencies:
   autoconf         noarch 2.69-27.el8
   perl-IO-Socket-IP noarch 0.39-5.el8
   ...  
   Installing weak dependencies:
   perl-IO-Socket-IP
   noarch 0.39-5.el8
   ...  
   Installing module profiles:
   php/common
   php/devel
   php/minimal
   Enabling module streams:
httpd                   2.4
nginx                   1.14
php                     7.3

Transaction Summary
========================================================================
=  Install  73 Packages
Total download size: 76 M
Installed size: 220 M
Is this ok [y/N]: y

2. Remove packages from the *devel* profile:

[root@rhel-8 ~]# yum module remove php:7.3/devel
Updating Subscription Management repositories.
Last metadata expiration check: 0:09:40 ago on Tue Mar  3 11:32:05 2020.
Dependencies resolved.
========================================================================
=  Package                Arch   Version                       Repository                         Size
========================================================================
=  Removing:
  libzip                 x86_64 1.5.2-1.module+el8.1.0+3189+a1bff096
                         @rhel-8-for-x86_64-appstream-rpms 313 k
  php-devel              x86_64 7.3.5-5.module+el8.1.0+4560+e0eee7d6
                         @rhel-8-for-x86_64-appstream-rpms 5.3 M
  php-pear               noarch 1:1.10.9-1.module+el8.1.0+3189+a1bff096
                         @rhel-8-for-x86_64-appstream-rpms 5.3 M
  php-pecl-zip           x86_64 1.15.4-1.module+el8.1.0+3189+a1bff096
                         @rhel-8-for-x86_64-appstream-rpms 2.1 M
  php-process            x86_64 7.3.5-5.module+el8.1.0+4560+e0eee7d6
                         @rhel-8-for-x86_64-appstream-rpms 119 k
Removing unused dependencies:
  autoconf               noarch 2.69-27.el8
                         @rhel-8-for-x86_64-appstream-rpms 2.2 M
...       Disabling module profiles:
  php/devel

Transaction Summary
========================================================================
=  Remove  64 Packages
Freed space: 193 M
Is this ok [y/N]: y

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
  ```

  The module is returned to the initial state. Information about an enabled stream and installed profiles is erased but 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 packages from an installed profile**

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

**Remove all packages from an active stream**

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

**Reset a module to the 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 to a later stream”** provides steps for changing to a later stream of a module.

### 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, follow the procedure in **Section 6.5, “Switching to a later stream”**. Alternatively, reset the module, and then enable the new stream. Removing all packages installed from a stream before switching to a different stream 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
  ```

  The module is returned to the initial state. Information about an enabled stream and installed profiles is erased but 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 TO A LATER STREAM
When you switch to a later module stream, all packages from the module are replaced with their later versions.

**IMPORTANT**

This procedure is feasible only under the conditions described below.

**Prerequisites**

- The system is fully updated.
- No packages installed on the system are newer than the packages available in the repository.

**Procedure**

1. Run the following command to determine if your system is prepared for switching to a later stream:

   ```bash
   # yum distro-sync
   ```

   This command must finish with the message *Nothing to do. Complete!* If it instead proposes changes and asks for confirmation, carefully review these changes, and consider whether you want to proceed. Run the `yum distro-sync` command repeatedly if necessary. Alternatively, you can refuse the suggested changes and manually modify your system to a state where the command returns *Nothing to do. Complete!*

   **NOTE**

   By checking the `yum distro-sync` result before switching the streams, you prevent making changes to the system that are unrelated to the stream switching because the same command is required as the last step of this procedure.

2. Change the active stream to the later one:

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

3. Synchronize installed packages to perform the change between streams:

   ```bash
   # yum distro-sync
   ```

   If this action suggests changes to content outside the streams, review them carefully.
NOTE

- If certain installed packages depend on the earlier stream and there is no compatible version in the later stream, `yum` will report a dependency conflict. In this case, use the `--allowerasing` option to remove such packages because they cannot be installed together with the later stream due to missing dependencies.

- When switching Perl modules, the `--allowerasing` option is always required because certain packages in the base RHEL 8 installation depend on Perl 5.26.

- Binary extensions (typically written in C or C++) for interpreted languages need to be reinstalled after the new stream is enabled; for example, certain packages installed by the `gem` command from the `ruby` module, the `npm` command from the `nodejs` module, the `cpan` command from the `perl` module, or the `pecl` command from the `php` module. For more information, see How to switch Ruby streams in RHEL 8.

Alternatively, remove all the module’s content installed from the current stream, reset the module, and install the new stream.