Oracle® Linux 8

Managing Software on Oracle Linux

ORACLE

F19386-19
November 2021

Abstract

Oracle® Linux 8: Managing Software on Oracle Linux provides information about how to install, upgrade, and manage software on Oracle Linux 8 systems by using DNF and Application Streams. Information is also provided on how to register with the Unbreakable Linux Network (ULN) and how to use this service to keep systems up to date and to access software that is not available in the repositories that are provided by the Oracle Linux yum server.

Document generated on: 2021-11-02 (revision: 12616)

Table of Contents

Preface

Oracle® Linux 8: Managing Software on Oracle Linux provides information about how to install, upgrade, and management software on your system by using DNF and Application Streams. Procedures for creating a local yum server, as well as instructions on using the `dnf` command is described. This guide also includes information about registering your systems with the Unbreakable Linux Network (ULN).

Audience

This document is intended for administrators who want to use the ULN. It is assumed that readers are familiar with web technologies and have a general understanding of Linux system administration.

Document Organization

The document is organized into the following chapters:

- Chapter 1, *Yum DNF* describes the `dnf` command that you use to install and update software on a system running Oracle Linux 8. Other useful commands are described, including how you can manage yum repositories using DNF plugins.

  This chapter also provides details about the Oracle Linux yum server, where Oracle makes open-source software available to Oracle Linux users. It also describes Application Streams and includes information about modules, including how to install them and how to use different versions of software on an Oracle Linux 8 system.

- Chapter 2, *Unbreakable Linux Network* describes the Unbreakable Linux Network (ULN) and how it works. Information about the packages that are required for a system to connect to ULN is also included, as well as how channels are named and how software errata are released to various channels, is also provided in this chapter.

  The following topics are also covered in this chapter: administering CSIs, ULN registration, channel management configuration and system management. The chapter also describes the application programming interface (API) that can be used to access and query ULN.

Related Documents

The documentation for this product is available at:

Oracle® Linux 8 Documentation

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit
https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Chapter 1 Yum DNF

Table of Contents

This chapter describes how to use the `yum` or `dnf` utility to install and upgrade software packages. Note that the `yum` command provided with Oracle Linux 8 is a symbolic link to the `dnf` command.

1.1 About DNF

The `yum` utility that is provided with Oracle Linux 8 is based on Dandified Yum (DNF). You can use `dnf` to install or upgrade RPM packages, while automatically handling package dependencies and requirements. The `dnf` command can be used to download the packages from repositories such as those that are available on the Oracle Linux yum server, but you can also set up your own repositories on systems that do not have Internet access. You can also use the `dnf` command on systems that are registered with the Unbreakable Linux Network (ULN) to install additional software that is limited to Oracle Linux Premier Support customers.

DNF provides significant improvements in functionality and performance when compared to the traditional `yum` command. DNF also brings a host of new features, including support for modular content and a more stable and well documented API. DNF is compatible with Yum v3, when used from the command line or when editing or creating configuration file. You can use the `dnf` command and all of its options similarly to how you used the `yum` command on previous releases of Oracle Linux.

The `yum` command that is provided with Oracle Linux 8 is a symbolic link to the `dnf` command. The commands are completely interchangeable. This implementation provides a level of backward compatibility that enables you to perform similar tasks to those that you performed in earlier releases of Oracle Linux, while at the same time, facilitating the wider range of new features that are available in `dnf`, such as improved package management and performance. Syntax differences between `dnf` and legacy `yum` commands are described in detail in *Oracle® Linux 8: Release Notes for Oracle Linux 8*.

1.1.1 About the Oracle Linux Yum Server

The Oracle Linux yum server is a convenient way to install Oracle Linux packages, including bug fixes, security fixes and enhancements, rather than installing them from installation media. You can access the server at https://yum.oracle.com/.

You can also subscribe to the Oracle Linux mailing list to be notified when new packages are released. You can access the mailing list at https://oss.oracle.com/mailman/listinfo/el-errata.

1.1.2 About Using ULN With Yum

The repositories that are available on the Oracle Linux yum server are aligned with the channels that are available from the Unbreakable Linux Network (ULN). Exceptions are any ULN channels that are limited to Oracle Linux Premier Support customers, for example, channels for products such as Ksplice.

Compute nodes running Oracle Linux on Oracle Cloud Infrastructure and that are connected to a service gateway automatically have access to ULN content via the regional yum servers available on the Oracle Services Network. These yum servers differ from the publicly available Oracle Linux yum server in that they also mirror content available on restricted ULN channels.

Access to ULN content is provided by virtue of the support contract that you have for your Oracle Cloud Infrastructure account. You are able to access content on ULN without any requirement to register or use alternate tools to manage channel access, simplifying any software management that you need to perform on a compute node.

To enable access to restricted content via the regional yum servers, ensure that you have installed the appropriate `release-el8` packages and enabled the repositories that you require access to.

1.2 DNF Configuration

The main configuration file for DNF is `/etc/dnf/dnf.conf`. The global definitions for DNF are located under the `[main]` section heading of the DNF configuration file. The following table describes the important directives for DNF.

Directive	Description
`cachedir`	Directory used to store downloaded packages.
`debuglevel`	Logging level, from 0 (none) to 10 (all).
`exclude`	A space separated list of packages to exclude from installs or updates, for example: `exclude=VirtualBox-4.? kernel*`.
`gpgcheck`	If set to 1, verify the authenticity of the packages by checking the GPG signatures. You might need to set `gpgcheck` to 0 if a package is unsigned, but you should be wary that the package could have been maliciously altered.
`gpgkey`	Path to the GPG public key file.
`installonly_limit`	Maximum number of versions that can be installed of any one package.
`keepcache`	If set to 0, remove packages after installation.
`logfile`	Path to the yum log file.
`obsoletes`	If set to 1, replace obsolete packages during upgrades.
`plugins`	If set to 1, enable plugins that extend the functionality of `yum`.
`proxy`	URL of a proxy server including the port number. See Section 1.2.1, “Configuring the Use of a Proxy Server”.

Directive	Description
`proxy_password`	Password for authentication with a proxy server.
`proxy_username`	User name for authentication with a proxy server.
`reposdir`	Directories where `yum` should look for repository files with a `.repo` extension. The default directory is `/etc/yum.repos.d`.

See the `dnf.conf(5)` manual page for more information.

The following listing shows an example `[main]` section from the DNF configuration file.

```
[main]
cachedir=/var/cache/dnf
keepcache=0
debuglevel=2
logfile=/var/log/dnf.log
obsoletes=1
gpgkey=file://media/RPM-GPG-KEY
gpgcheck=1
plugins=1
installonly_limit=3
```

It is possible to define repositories below the `[main]` section in `/etc/dnf/dnf.conf` or in separate repository configuration files. By default, `dnf` expects any repository configuration files to be located in the `/etc/yum.repos.d` directory, unless you use the `reposdir` directive to define alternate directories.

Note that for backward-compatibility purposes, a symbolic link to `/etc/dnf/dnf.conf` is created at `/etc/yum.conf`. The configuration syntax is generally the same; although, some configuration options have been deprecated and some new configuration options have been added. See *Oracle® Linux 8: Release Notes for Oracle Linux 8* for a breakdown of the differences between configuration options and syntax.

1.2.1 Configuring the Use of a Proxy Server

If your organization uses a proxy server as an intermediary for Internet access, specify the `proxy` setting in `/etc/dnf/dnf.conf` as shown in the following example.

```
proxy=http://proxysvr.example.com:3128
```

If the proxy server requires authentication, additionally specify the `proxy_username`, and `proxy_password` settings.

```
proxy=http://proxysvr.example.com:3128
proxy_username=user
proxy_password=password
```

If you use the yum plugin (`yum-rhn-plugin`) to access the ULN, specify the `enableProxy` and `httpProxy` settings in `/etc/sysconfig/rhn/up2date` as shown in this example.

```
enableProxy=1
httpProxy=http://proxysvr.example.com:3128
```

If the proxy server requires authentication, additionally specify the `enableProxyAuth`, `proxyUser`, and `proxyPassword` settings.

```
enableProxy=1
httpProxy=http://proxysvr.example.com:3128
enableProxyAuth=1
proxyUser=user
proxyPassword=password
```

Caution

All dnf users require read access to /etc/dnf/dnf.conf or /etc/sysconfig/rhn/up2date. If these files must be world-readable, do not use a proxy password that is the same as any user's login password, and especially not root's password.

1.2.2 Yum Repository Configuration

Yum repository configuration files are used by DNF to determine where different packages and their dependencies can be installed from.

The yum configuration file or yum repository configuration files can contain one or more sections that define repositories.

The following table describes the basic directives for a repository.

Directive	Description
baseurl	Location of the repository channel (expressed as a file://, ftp://, http://, or https:// address). This directive must be specified.
enabled	If set to 1, permit yum to use the channel.
name	Descriptive name for the repository channel. This directive must be specified.

Any other directive that appears in this section overrides the corresponding global definition in the [main] section of the DNF configuration file. See the dnf.conf(5) manual page for more information.

The following listing shows an example repository section from a configuration file.

```
[ol8_appstream]
name=Oracle Linux $releasever Application Stream ($basearch)
baseurl=https://yum.oracle.com/repo/OracleLinux/OL8/appstream/$basearch
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1
```

In this example, the values of gpgkey and gpgcheck override any global setting. dnf substitutes the name of the current system's architecture for the variable *$basearch*.

The dnf command automatically searches the /etc/yum.repos.d directory for files with the suffix .repo and appends these to the configuration when it is processing. Use this directory to define repository files for repositories that you want to make available.

1.2.3 Downloading Oracle Linux Yum Server Repository Files

The Oracle Linux yum server provides a direct mapping of all of the ULN channels that are available to the public, without any specific support agreement. The repository labels that are used for each repository on the Oracle Linux yum server map directly to the channel names on ULN. See Chapter 2, *Unbreakable Linux Network* for more information about channel names and common suffixes that are used for channels and repositories.

Oracle Linux 8 uses modular yum repository configuration files released as packages that can be maintained through yum, which helps simplify repository management and also ensure that your yum repository definitions are kept up to date automatically whenever you update your system.

A list of all available RPM files to manage all of the possible yum repository configurations for your release can be obtained by running the following command:

```
dnf list "*release-el8*"
```

To install the yum repository configuration for a particular set of software that you wish to use, use the `dnf` command to install the corresponding package.

If, for some reason, you manage to remove all configuration to access the Oracle Linux yum server repositories, you should create a temporary yum repository configuration file at `/etc/yum.repos.d/ol8-temp.repo` with the following as the minimum required content:

```
[ol8_baseos_latest]
name=Oracle Linux $releasever BaseOS ($basearch)
baseurl=https://yum$ociregion.oracle.com/repo/OracleLinux/OL8/baseos/latest/$basearch
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1
```

Then, reinstall the `oraclelinux-release-el8` package to restore the default yum configuration:

```
sudo dnf reinstall oraclelinux-release-el8
sudo rm /etc/yum.repos.d/ol8-temp.repo
```

For more information about manually setting up Oracle Linux yum server repository configuration files, see https://yum.oracle.com/getting-started.html.

You can enable or disable repositories in each repository configuration file by setting the value of the `enabled` directive to `1` or `0`, for each repository that is listed in the file, as required. The preferred method of enabling or disabling repositories under Oracle Linux 8 is to use the `dnf config-manager` command, as described in Section 1.2.4, "Using the DNF config-manager Plugin".

1.2.3.1 Configuring Oracle Cloud Infrastructure Compute Instances for access to the regional yum server repositories

Compute instances in Oracle Cloud Infrastructure have access to regional yum servers via the service gateway. Regional yum servers on Oracle Cloud Infrastructure differ from the Oracle Linux yum server in that they also mirror content available on restricted ULN channels.

Yum repository configuration in Oracle Linux makes use of a yum variable in the baseurl for managing appropriate yum server access. For example, the baseurl to the `ol8_baseos_latest` repository for Oracle Linux 8 is:

```
baseurl=https://yum$ociregion.oracle.com/repo/OracleLinux/OL8/baseos/latest/$basearch
```

The `$ociregion` variable can be set by populating content in `/etc/dnf/vars/ociregion`. If this file does not exist, or the file is empty, the baseurl is expanded to point to the publicly accessible Oracle Linux yum server. In the case of a typical Oracle Cloud Infrastructure compute instance, the value of variable is set when the instance is created so that the baseurl is expanded to point to the closest regional yum server on the Oracle Cloud Infrastructure service network. For example, if `$ociregion` is set to `-phx`, the baseurl expands to point to the regional yum server located in Phoenix.

By using variables, configuration can remain relatively standard across Oracle Linux deployments but provide access to the additional resources available to Oracle Cloud Infrastructure customers.

1.2.4 Using the DNF config-manager Plugin

The `dnf-plugins-core` package includes several utilities that can help you to manage configuration and safely apply updates to your existing configuration. Most significant of these is the `dnf config-manager` plugin.

You can use `dnf config-manager` to add repositories, either at a specified URL or within a specified repository file. For example, to add a repository configuration file for Oracle Linux 8 that is hosted on a remote server, you can run the following command:

```
sudo dnf config-manager --add-repo https://example.com/my_yum_config.repo
```

You can use the same command to automatically generate a repository configuration file for a valid yum repository by pointing to the URL for which the repository is hosted. For example, to create a new configuration file in `/etc/repos.d` for an example repository, run the following command:

```
sudo dnf config-manager --add-repo https://example.com/repo/OracleLinux/OL8/myrepo/x86_64
```

To enable a repository by using `dnf config-manager`, use the `--enable` option. For example, to enable the `ol8_appstream` repository, run the following command:

```
sudo dnf config-manager --enable ol8_appstream
```

You can use the `--disable` option in a similar way to disable a repository.

You can also use the `dnf config-manager` tool to set other configuration options by specifying the `--setopt` and `--save` options. See the `dnf.plugin.config_manager(8)` manual page for more information.

1.3 Using DNF From the Command Line

The following table shows examples of some of the common tasks that you can perform by using the `dnf` command.

Command	**Description**
`dnf repolist`	Lists all of the enabled repositories.
`dnf list`	Lists all of the packages that are available in all enabled repositories and all packages that are installed on your system.
`dnf list installed`	Lists all of the packages that are installed on your system.
`dnf list available`	Lists all of the packages that are available to be installed in all enabled repositories.
`dnf search` *`string`*	Searches the package descriptions for the specified string.
`dnf provides` *`feature`*	Finds the name of the package to which the specified file or feature belongs, for example: `dnf provides /etc/dnf/automatic.conf`
`dnf info` *`package`*	Displays detailed information about a package, for example: `dnf info dnf-automatic`
`dnf repoquery -l` *`package`*	List the files that are contained in a package and are installed when the package is installed, for example: `dnf repoquery -l dnf-automatic`
`dnf install` *`package`*	Installs the specified package, including packages on which it depends, for example: `dnf install dnf-automatic`

Command	Description
`dnf check-update`	Checks whether updates exist for packages that are already installed on your system.
`dnf upgrade` *`package`*	Updates the specified package, including packages on which it depends, for example: `dnf upgrade dnf-automatic` DNF also interprets the `dnf update` *`package`* command as synonymous with the upgrade syntax; however, this syntax is considered deprecated.
`dnf upgrade`	Updates all packages, including packages on which they depend. DNF also interprets the `dnf update` *`package`* command as synonymous with the upgrade syntax; however, this syntax is considered deprecated.
`dnf remove` *`package`*	Removes the specified package. For example: `dnf remove dnf-automatic`
`dnf clean all`	Removes all cached package downloads and cached headers that contain information about remote packages. Running this command can help to clear problems that are a result of unfinished transactions or out-of-date headers.
`dnf help`	Displays help about `dnf` usage.
`dnf help` *`command`*	Displays help about the specified `dnf` command, for example: `dnf help upgrade`
`dnf shell`	Runs the `dnf` interactive shell.

See the `dnf(8)` manual page for more information.

Note

`dnf` makes no distinction between installing and upgrading a kernel package. `dnf` always installs a new kernel regardless of whether you specify `upgrade` or `install`.

1.4 DNF Groups

A set of packages can be organized and managed as a *group*. Groups can be nested so that a parent group contains a set of sub-groups that can be installed. Examples include the groups for setting up a virtualization host, a graphical desktop, a collection of fonts, or core system administration tools. The following table shows the `dnf` commands that you can use to manage these groups.

Command	Description
`dnf group list`	Lists Environment Groups, that contain many sub-groups; and base groups of packages that are available for installation.
`dnf group info` *`groupname`*	Displays detailed information about a group. If the group is a parent group, this command lists all sub-groups that it contains, alternately the command lists all packages that are in the group.
`dnf group install` *`groupname`*	Installs all of the packages in a group.

Command	Description
`dnf group update` *`groupname`*	Updates all of the packages in a group.
`dnf group remove` *`groupname`*	Removes all of the packages in a group.

1.5 DNF Modules and Application Stream

DNF introduces the concepts of modules, streams and profiles to allow for the management of different versions of software applications within a single operating system release. Modules can be used to group together many packages that comprise a single application and its dependencies. Streams can be used to provide alternate versions of the same module. Profiles can be used to define optional configurations of any single module so that a module can be limited only to developer packages or can be scoped to include additional packages for enhanced functionality.

Modular content is made available separately to core operating system packages so that these user-space applications can be installed in a variety of user-space environments, including virtual machines, containers as well as the base operating system. Modular content for Oracle Linux 8 is typically shipped within the Application Stream (AppStream) repository. For a list of application stream packages in the latest Oracle Linux 8 version, see Appendix A, *Application Stream Module Life Cycle*.

- **Modules**: Are a set of RPM packages that are grouped together and must be installed together. They can contain several streams that consist of multiple versions of applications that you can install. You enable a module stream to provide system access to the RPM packages that are contained in that module stream.

  A typical module can contain the following types of packages:

  - Packages with an application
  - Packages with the application's specific dependency libraries
  - Packages with documentation for the application
  - Packages with helper utilities

- **Module streams**: Hold different versions of content contained within a module.

  Modules can have multiple streams, where each stream contains a different version of packages and their dependencies. Each stream receives updates independently. A module can have more than one stream. However, note that for each module, only one of its streams can be enabled to provide access to its packages. Frequently, the stream with the latest version is selected as the default stream and is used when operations do not specify a particular stream or a different stream has not been enabled previously.

  Module streams can be thought of as virtual repositories within the physical repository. For example, the `postgresql` module provides the PostgreSQL database, in streams 9.6 and 10, respectively, with version 10 being the current default stream.

  Note

  Oracle recommends that you use the latest stream for any module that is installed, even though other streams may continue to receive limited support.

- *Module profiles*: Provide a list of certain packages that are to be installed at the same time for a particular use case. At the same time, profiles are also a recommendation by the application packagers and experts. Note that each module can have one or more profiles.

You install packages by using a module's profile as a one-time action. Using a module's profile to install packages does not prevent you from installing or uninstalling any of the packages that are provided by the module. Furthermore, it is possible to install packages by using multiple profiles of the same module without any further preparation. Also, a module's package list can contain packages from outside of the module stream, usually from BaseOS or stream's dependencies. Note that modules in Application Stream always have a default profile. This default profile is used for installations, when no other profile has been explicitly specified.

For example, The `httpd` module that includes the Apache web server supports the following profiles for installation:

- `common`: This profile is a hardened production-ready deployment and is the default profile.
- `devel`: This profile installs the packages that are necessary to make modifications to `httpd`.
- `minimal`: This profile installs the smallest set of packages that provide a running web server.

Unlike software collections that were included in previous releases of Oracle Linux, applications that are installed from Application Streams are installed into standard locations and do not require additional commands or actions to run. You can run any version of an installed application the same way as any other version, regardless of the stream from which it was installed. After it is installed, the application behaves exactly as any other native application that you have installed by using DNF.

1.5.1 Displaying Available Modules

You can list available modules, typically within the Application Stream repository, by using the `dnf module list` command. Further module information can be obtained using the `dnf module info` command.

The following table describes some of the more commonly used commands for viewing and displaying content details in Application Stream.

Command Syntax	**Description of Action**
`dnf module info` *`module-name`*	Displays information about a module.
`dnf module info --profile` *`module-name`*	Displays information about the packages that are installed by the profiles of a module using the *default* stream.
`dnf module info --profile` *`module-name`*:*`stream`*	Displays information about the packages that are installed by the profiles of a module using a *specified* stream.
`dnf module list`	Lists all of the available modules and displays the module name, stream, profiles, and a summary. Each module and stream is listed on a separate line. Profiles are indicated using comma separated values for each module and stream. Default values are indicated with the characters `[d]`. Modules that are enabled are indicated with the characters `[e]`, while those that are disabled are indicated with the characters `[x]`. Installed modules, streams and profiles are indicated with the characters `[i]`.
`dnf module list` *`module-name`*	Lists the current status of a module.
`dnf module provides` *`package`*	Displays information about which modules provide a specified package.

Command Syntax	Description of Action
	If the package is only available outside of any modules, the command output is empty.

1.5.2 Module Installation Commands

The following table describes the commands that are used to install content from Application Stream.

Command Syntax	Description of Action	Additional Information
`dnf install` *`package`*	Installs the specified package.	If a package is provided by a module stream, the `dnf` command resolves the required module stream and enables it automatically during package installation. In addition, the process is recursive for any package dependencies. Note that if more module streams satisfy the requirement, the default streams are used. If the package is provided by a module stream that is not marked as default or is not enabled, that package is not recognized until you manually enable the applicable module stream.
`dnf module enable` *`module-name`*:*`stream`*	Enables a module or stream.	Use this command when you want to enable a module so that the packages are available to the system, but you do not necessarily want to install the module immediately. Note that some modules might not define default streams. In this case, you must explicitly specify the stream. If you explicitly specify a stream and an alternate stream is set as the default, the enabled stream overrides the default stream for subsequent install requests.

`dnf install` *`@module-name`*
 	
Alternatively, you can use:
 	
`dnf module install` *`module-name`*
	Installs a module. The `@` character is shorthand to indicate that you intend to install a module.	If the module defines a default stream, or you have enabled a particular stream, you do not need to include *`stream`* and *`colon`* in the command syntax.
		
		Be aware that some modules do not define default streams.

`dnf install` *`@module-name`*:*`stream`*	Installs a module by using a specific stream and default profiles.	

Command Syntax	Description of Action	Additional Information
Alternatively, you can use: `dnf module install` *module-name*:*stream*		
`dnf install` *@module-name*:*stream*/*profile* Alternatively, you can use: `dnf module install` *module-name*:*stream*/*profile*	Installs a module by using a specific stream and profile.	

1.5.3 About Modular Dependencies and Stream Changes

Typically, packages that provide content depend on other packages, and they usually specify the desired dependency versions. This same mechanism also applies to packages that are contained within modules. The grouping of packages and their particular versions into modules and streams has some additional constraints. For example, module streams can declare dependencies on the streams of other modules, independent of the packages that are contained and provided by them. After any package or module operation, the entire dependency tree for all of the underlying installed packages must satisfy all of the conditions that the packages declare. Also, all of the module stream dependencies must satisfied.

These additional constraints require that you carefully consider any package operations prior to performing them, as changing the enabled module streams does not automatically manipulate packages to enable you to have complete control over the changes. However, the tool always provides a summary of the actions to take.

When performing package operations on modules and streams, keep the following guidelines, caveats, and warnings in mind:

- Enabling a module stream might also require the enabling of streams of additional modules.
- Installing a module stream profile or packages from a stream might also require the enabling of streams of additional modules and the installation of additional packages.
- Disabling a stream of a module might also require the disabling of other module streams, as no packages are removed automatically.
- Removing a package can require the removal of additional packages. If any of the packages are provided by modules, the module streams remain enabled in preparation for further installation, even if no packages from these streams are installed subsequently; thereby, mirroring the behavior of an unused yum repository.
- Switching the stream that is enabled for a module is the same as resetting the current stream and enabling a new stream.

  Note

  Switching an enabled stream does not automatically change any of the installed packages. Also, removing packages that are provided by a previous stream, and any of the packages that depend on them, as well as the installation of packages in a new stream are all tasks that must be performed manually.

- Due to potential upgrade scripts that run during an installation, directly installing a stream of a module, other than one that is currently installed by default, is not recommended.

Module dependencies include regular package dependencies that are similar to RPM dependencies. For modules, however, availability can also depend on the enabling of module streams; module streams can also depend on other module streams.

Dependencies of non-modular packages on modular packages is used in Application Stream only when a modular package is provided by a module stream that is marked as the default. When a modular package depends on a non-modular package, the system always retains the module and stream choices, unless you provide explicit instructions to change them. A modular package receives updates from the currently enabled stream of the module that provides this package and does not upgrade to a version from a different stream.

1.5.4 Removing Installed Modules

Before removing an installed module, carefully review the information in Section 1.5.3, "About Modular Dependencies and Stream Changes".

When you remove an installed module, all of the packages that are installed by the profiles of the currently enabled module stream, and any further packages and modules that depend on them, are also removed. Note that any packages installed from this module stream that are not listed in any of its profiles remain installed on the system and can be removed manually.

Note

A prerequisite to removing installed modules requires that the module to be removed already has some profiles installed.

To remove an installed module, follow these steps:

1. Remove the module.

   ```
   sudo dnf module remove module-name
   ```

   In the previous example, *`module-name`* specifies the name of the module to remove.

   The `dnf module remove` command removes all of the packages that are installed from this module. You are presented with a summary of the changes to be made and a request for confirmation.

2. Disable the module stream.

   ```
   sudo dnf module disable module-name
   ```

   In the previous example, *`module-name`* is the name of the module to disable.

   You are presented with a summary of the changes to be made and a request for confirmation.

3. Remove any packages that you manually installed from the module stream.

   ```
   sudo dnf remove package ...
   ```

   In the previous example, *`package ...`* is the name of the package, or packages, to be removed.

   You are presented with a summary of the changes to be made and a request for confirmation.

1.5.5 Switching Module Streams

Before switching module streams, carefully review the information in Section 1.5.3, "About Modular Dependencies and Stream Changes".

When you switch to a different module stream, you are usually upgrading or downgrading the content to a different version than the version that is currently installed on the system.

Note

The module stream that you want to switch must already be enabled, *and*, another stream of the same module must already exist.

1. Reset the module to make it possible to install an alternate stream:

   ```
   sudo dnf module reset module-name
   ```

   If you fail to reset the module before attempting to install an alternate stream, an error is returned to notify you that it is not possible to switch enabled streams of a module.

2. Install the profiles of a different stream of the module as follows:

   ```
   sudo dnf install @module-name:stream
   ```

   In the previous example, *`module-name`* is the name of the module and *`stream`* is the desired stream.

   You are presented with a summary of the changes to be made and a request for confirmation.

   Running the previous command enables the new stream and disables the current stream. Note that it might be necessary to make changes to additional module streams and packages.

3. Update or downgrade any packages installed from the previous module stream that were not listed in the profiles installed in the previous step.

   ```
   sudo dnf distro-sync
   ```

   You are presented with a summary of the changes to be made and a request for confirmation.

4. Manually remove any remaining packages that were installed from the previous module stream.

   ```
   sudo dnf remove package ...
   ```

   In the previous example, *`package ...`* is the name of the package, or packages, to be removed.

   You are presented with a summary of the changes to be made and a request for confirmation.

1.6 Using DNF Security Options

DNF includes integrated options to handle any requirement for managing security and errata updates that are available for packages installed in Oracle Linux 8.

List the errata that are available for your system as follows:

```
sudo dnf updateinfo list
...
```

The output from the command sorts the available errata in order of their IDs, and it also specifies whether each erratum is a security patch (*`severity/`*`Sec.`), a bug fix (`bugfix`), or a feature enhancement (`enhancement`). Security patches are listed by their severity: `Important`, `Moderate`, or `Low`.

You can use the `--sec-severity` option to filter the security errata by severity, for example:

```
sudo dnf updateinfo list --sec-severity=Moderate
...
```

To list the security errata by their Common Vulnerabilities and Exposures (CVE) IDs instead of their errata IDs, specify the keyword `cves` as an argument:

```
sudo dnf updateinfo list cves
...
```

Similarly, the keywords `bugfix`, `enhancement`, and `security` filter the list for all bug fixes, enhancements, and security errata.

You can use the `--cve` option to display the errata that correspond to a specified CVE, for example:

```
sudo dnf updateinfo list --cve CVE-2020-4000
```

To display more information, specify `info` instead of `list`, for example:

```
sudo dnf updateinfo info --cve CVE-2020-4000
```

To update all of the packages for which security-related errata are available to the latest versions of the packages, even if those packages that include bug fixes or new features but not security errata, use the following command:

```
sudo dnf --security update
```

To update all packages to the latest versions that contain security errata, ignoring any newer packages that do not contain security errata, use the following command:

```
sudo dnf --security upgrade-minimal
```

To update all kernel packages to the latest versions that contain security errata, use the following command:

```
sudo dnf --security upgrade-minimal kernel*
```

To update only those packages that correspond to a CVE or erratum, use the `dnf update --cve` and `dnf update --advisory`commands, for example:

```
sudo dnf update --cve CVE-2020-4000
```

```
sudo dnf update --advisory ELSA-2020-4010
```

Note

Some updates might require that you reboot the system. By default, the boot manager automatically enables the most recent kernel version.

For more information, see the `dnf(8)` manual page.

1.7 Using the DNF Automatic Tool to Keep Your System Up To Date

The DNF Automatic tool is provided as an additional package that you can use as an alternative to manually running `dnf upgrade` to keep your system up to date with the latest security patches and bug fixes. The tool can provide automatic notifications of updates, download updates, and then install them automatically by using systemd timers.

You can install the `dnf-automatic` package and enable the systemd `dnf-automatic.timer` timer unit to start using this service:

```
sudo dnf install dnf-automatic
```

```
sudo systemctl enable --now dnf-automatic.timer
```

You configure the DNF Automatic tool by editing the `/etc/dnf/automatic.conf` configuration file and then restarting the timer unit.

Note that additional alternate timer units are available and can override the default configuration that is specified in the configuration file. Frequently, these timer units are used as handy shortcuts to perform a specific behavior:

- `dnf-automatic-notifyonly.timer`: Notifies for available updates
- `dnf-automatic-download.timer`: Downloads package updates, but does not install them
- `dnf-automatic-install.timer`: Downloads and automatically installs package updates

You enable the required behavior by running:

```
sudo systemctl enable --now dnf-automatic-install.timer
```

1.8 Creating a Local Yum Repository by Using an ISO Image

Note

The system must have sufficient storage space to host a full Oracle Linux Media Pack DVD image (approximately 6.6 GB for Oracle Linux 8).

To create a local yum repository (for example, if a system does not have Internet access):

1. On a system with Internet access, download a full Oracle Linux DVD image from the Oracle Software Delivery Cloud at https://edelivery.oracle.com/linux onto removable storage (such as a USB memory stick).

   Note

   You can verify that the ISO was copied correctly by comparing its checksum with the digest value that is listed on `edelivery.oracle.com`, for example:

   ```
   sudo sha1sum OracleLinux8.iso
   ```

   ```
   203b8185d8c6551378b41da26b088f23e131343f OracleLinux8.iso
   ```

2. Transfer the removable storage to the system on which you want to create a local yum repository, and copy the DVD image to a directory in a local file system.

   ```
   sudo cp /media/USB_stick/OracleLinux8.iso /ISOs
   ```

3. Create a suitable mount point, for example `/var/OSimage/OL8_x86_64`, and mount the DVD image on it.

   ```
   sudo mkdir -p /var/OSimage/OL8_x86_64
   ```

   ```
   sudo mount -o loop,ro /ISOs/OracleLinux8.iso /var/OSimage/OL8_x86_64
   ```

   Note

   Include the read-only mount option (`ro`) to avoid changing the contents of the ISO by mistake.

4. Create an entry in `/etc/fstab` so that the system always mounts the DVD image after a reboot.

   ```
   /ISOs/OracleLinux8.iso /var/OSimage/OL8_x86_64 iso9660 loop,ro 0 0
   ```

5. Disable all existing yum repositories.

   In the `/etc/yum.repos.d` directory, edit any existing repository files and disable all entries by setting `enabled=0`. Alternately, as described in Section 1.2.4, “Using the DNF config-manager Plugin”, you can disable all repositories by running:

   ```
   sudo dnf config-manager --disable \*
   ```

6. Create the following entries in a new repository file (for example, `/etc/yum.repos.d/OL8.repo`), for example:

   ```
   [OL8_BaseOS]
   name=Oracle Linux 8 x86_64 ISO  BaseOS
   baseurl=file:///var/OSimage/OL8_x86_64/BaseOS
   gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
   gpgcheck=1
   enabled=1

   [OL8_AppStream]
   name=Oracle Linux 8 x86_64 ISO AppStream
   baseurl=file:///var/OSimage/OL8_x86_64/AppStream
   gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
   gpgcheck=1
   enabled=1
   ```

   Note that the correct GPG key file must exist at the path specified for the `gpgkey` parameter. You can download the GPG keys used to sign all of the Oracle Linux release packages from the Oracle Linux yum server, but the key is also provided on the ISO itself. See https://yum.oracle.com/faq.html#a10 for more information.

7. Clean up the dnf cache.

   ```
   sudo dnf clean all
   ```

8. Test that you can use the `dnf` command to access the repository.

   ```
   sudo dnf repolist
   ```

   ```
   Loaded plugins: refresh-packagekit, security
   ...
   repo id                          repo name                               status
   OL8_AppStream                    Oracle Linux 8 x86_64 ISO AppStream     5,783
   OL8_BaseOS                       Oracle Linux 8 x86_64 ISO  BaseOS       1,697
   repolist: 7,480
   ```

1.9 Setting up a Local Yum Server by Using an ISO Image

To set up a local yum server (for example, if you have a network of systems that do not have Internet access):

1. Choose one of the systems to be the yum server, and create a local yum repository on it as described in Section 1.8, “Creating a Local Yum Repository by Using an ISO Image”.

2. Install the Apache HTTP server from the local yum repository.

   ```
   sudo dnf install httpd
   ```

3. If SELinux is enabled in enforcing mode on your system, do the following:

   a. Use the `semanage` command to define the default file type of the repository root directory hierarchy as `httpd_sys_content_t`:

```
sudo /usr/sbin/semanage fcontext -a -t httpd_sys_content_t "/var/OSimage(/.*)?"
```

b. Use the `restorecon` command to apply the file type to the entire repository:

```
sudo /sbin/restorecon -R -v /var/OSimage
```

Note

The `semanage` and `restorecon` commands are provided by the `policycoreutils-python` and `policycoreutils` packages.

4. Create a symbolic link in `/var/www/html` that points to the repository:

```
sudo ln -s /var/OSimage /var/www/html/OSimage
```

5. Edit the HTTP server configuration file (`/etc/httpd/conf/httpd.conf`) as follows:

   a. Specify the resolvable domain name of the server, in the argument to `ServerName`.

   ```
   ServerName server_addr:80
   ```

   If the server does not have a resolvable domain name, enter its IP address instead.

   b. Verify that the setting of the `Options` directive in the `<Directory "/var/www/html">` section specifies `Indexes` and `FollowSymLinks` to allow you to browse the directory hierarchy, for example:

   ```
   Options Indexes FollowSymLinks
   ```

   c. Save your changes to the file.

6. Start the Apache HTTP server, then configure it to start after a reboot.

```
sudo systemctl start httpd
```

```
sudo systemctl enable httpd
```

7. If you have enabled a firewall on your system, configure it to allow incoming HTTP connection requests on TCP port 80, for example:

```
sudo firewall-cmd --zone=zone --add-port=80/tcp
```

```
sudo firewall-cmd --permanent --zone=zone --add-port=80/tcp
```

8. Disable all of the existing yum repositories on the server and each client system.

   In the `/etc/yum.repos.d` directory, edit any existing repository files and disable all entries by setting `enabled=0`. If you have the `yum-utils` package installed, as described in Section 1.2.4, “Using the DNF config-manager Plugin”, you can disable all repositories by running the following command:

```
sudo dnf config-manager --disable \*
```

9. Edit the repository file on the server, for example, `/etc/yum.repos.d/OL8.repo`:

```
[OL8_BaseOS]
name=Oracle Linux 8 x86_64 ISO BaseOS
baseurl=http://server_addr/OSimage/OL8_x86_64/BaseOS
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1
```

```
[OL8_AppStream]
name=Oracle Linux 8 x86_64 ISO AppStream
baseurl=http://server_addr/OSimage/OL8_x86_64/AppStream
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1
```

Replace *server_addr* with the IP address or resolvable host name of the local yum server.

10. On the server and on each client, copy the repository file and copy and import the GPG key file that is used to sign these packages. You can either import the key included on the ISO, or you can download and install it from the Oracle Linux yum server.

```
sudo wget https://yum.oracle.com/RPM-GPG-KEY-oracle-ol8 -O /etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
```

```
sudo gpg --import --import-options show-only /etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
```

11. On each client, copy the repository file from the server to the `/etc/yum.repos.d` directory.

12. On the server and each client, test that you can use the `dnf` command to access the repository:

```
sudo dnf repolist
```

```
Loaded plugins: refresh-packagekit, security
...
repo id                         repo name                                status
OL8_AppStream                   Oracle Linux 8 x86_64 ISO AppStream      5,783
OL8_BaseOS                      Oracle Linux 8 x86_64 ISO  BaseOS        1,697
repolist: 7,480
```

1.10 More Information About DNF

More information on DNF is available at https://dnf.readthedocs.io/en/latest/index.html.

Frequently asked questions about the Oracle Linux yum server are answered at https://yum.oracle.com/faq.html.

Chapter 2 Unbreakable Linux Network

Table of Contents

This chapter describes the Unbreakable Linux Network (ULN) and explains how it works. The chapter provides a description of the packages that are required for a system to connect to ULN and also describes how channels are named and how software errata are released to different channels.

2.1 About the Unbreakable Linux Network

If you have a subscription to Oracle Unbreakable Linux support, you can use the comprehensive resources of ULN. ULN offers software patches, updates, and fixes for Oracle Linux and Oracle VM, as well as information about `yum`, `dnf`, Ksplice, and support policies. You can also download useful packages that are not included in the original distribution. The ULN Alert Notification Tool periodically checks with ULN and alerts you when updates are available. You can access ULN at https://linux.oracle.com/, where you will also find instructions for registering with ULN and creating local yum repositories.

If you want to use `dnf` with ULN to manage your systems, you must register them with ULN and subscribe each system to one or more ULN channels. When you register a system with ULN, the channel that contains the latest version is chosen automatically, according to the system's architecture and operating system revision. See Section 2.3, “ULN Registration” for more information.

When you run the `dnf` command, it connects to the ULN server repository and downloads the latest software packages onto your system in RPM format. `dnf` then presents you with a list of the available packages so that you can choose which packages you want to install.

2.1.1 About the `rhn-setup` Package

The tools to register with ULN from an Oracle Linux system are provided in the `rhn-setup` package. This package is available in the `ol8_appstream` yum repository that is available on the Oracle Linux yum server and which is included on the Oracle Linux 8 Installation ISO. This package is usually also installed by default on a new installation of an Oracle Linux 8 system.

You can also manually download the RPMs from ULN, directly, by browsing the appropriate channel and architecture for your system.

2.1.2 About ULN Channels

ULN provides more than 100 unique channels, which support the i386, x86_64, IA64, and the 64-bit Arm architectures, for releases of Oracle Linux 4 update 6 and later and Oracle VM 2.1 and later.

You can choose that your system remain at a specific operating system revision, or you can allow the system to be updated with packages from later revisions.

You should subscribe to the channel that corresponds to the architecture of your system and the update level at which you want to maintain it. Patches and errata are available for specific revisions of Oracle Linux, but you do not need to upgrade from a given revision level to install these fixes. ULN channels also exist for MySQL, Oracle VM, Oracle Ksplice, OCFS2, RDS, and productivity applications.

The following table describes the main channels that are available for Oracle Linux 8.

Channel	Description
ol8_*arch*_baseos_latest	Provides all the latest versions of the base operating system packages in the distribution, including any errata. If no vulnerabilities have been found in a package, the package version might be the same as that included in the original distribution. For other packages, the version is set at the highest update level.
ol8_*arch*_appstream	Provides all the latest versions of the Application Stream user space packages in the distribution, including any errata. If no vulnerabilities have been found in a package, the package version might be the same as that included in the original distribution. For other packages, the version is set at the highest update level.
ol8_*arch*_u*n*_baseos_base	Provides the base versions of the of the base operating system packages in the distribution as they are when they are release when a particular update level is released. Note that in the case of the initial release of Oracle Linux 8, *n* has a value of 0. Errata patches are not provided in this channel. If you want to keep your system up to date and secure, you should also subscribe to the appropriate _baseos_patch channel or subscribe to the appropriate _baseos_latest channel.
ol8_*arch*_u*n*_baseos_patch	Provides the patched versions of the of the base operating system packages in the distribution as they are when they are release when a particular update level is released. As errata patches are made available, the updates are released into this channel. Note that in the case of the initial release of Oracle Linux 8, *n* has a value of 0. Errata patches are provided in this channel until such time as a new update release is made available. If you want to keep your system up to date and secure, you should subscribe to the appropriate _baseos_latest channel.
ol8_*arch*_addons	Provides packages released by Oracle in addition to the upstream packages made available in the other channels listed here. These packages are usually specific to functionality that Oracle provides to improve user experience on Oracle Linux and to provide access to services specific to Oracle.
ol8_*arch*_oci	Provides packages specific to Oracle Cloud Infrastructure customers. The packages in this channel should only be used on compute instances in Oracle Cloud Infrastructure. This channel is available on ULN and is mirrored to the regional yum servers within the Oracle Cloud Infrastructure, but is not mirrored to the publicly accessible Oracle Linux yum server.
ol8_*arch*_codeready_builder	Provides the packages released in the upstream codeready_builder channel. The packages released in this channel are intended for developers who intend to build binary content from source packages. The packages include compilers, libraries and source required for package building and other related tasks. Many of the packages in this channel have dependencies on packages in the ol8_*arch*_appstream channel. Content in this channel is not intended for production use and is unsupported by Oracle.
ol8_*arch*_developer	Provides packages intended for developers to create test and development environments for Oracle Linux 8 and related technologies. Content in this channel is not intended for production use and is unsupported by Oracle.

Other channels may also become available, such as channels for the beta versions of packages, or for specific developer content.

Not all channels are available for all architectures. You can check what channels are available for your architecture in the ULN Web Interface. See Section 2.4.1, “ULN Channel Subscription Management” for more information.

As each new, major version or minor update of Oracle Linux becomes available, Oracle creates new base and patch channels for each supported architecture to distribute new packages. The existing base and patch channels for the previous versions or updates remain available and do not include the new packages. The `_latest` channel distributes the latest possible version of any package, and tracks the top of the development tree independently of the update level.

> **Caution**
>
> You can choose to maintain your system at a specific update level of Oracle Linux and selectively apply errata to that level by subscribing the system to the `_base` and `_patch` channels and unsubscribing it from the `_latest` channel. However, patches are not added to the `_patch` channel for previous updates of Oracle Linux after a new update has been released. For example, after the release of Oracle Linux 8 Update 1, no further errata will be released on the `ol8_x86_64_u0_baseos_patch` channel.
>
> Oracle recommends that you keep you system subscribed to the `_latest` channel. If you unsubscribe from the `_latest` channel, your system will become vulnerable to security-related issues when a new update is released.

For more information about the channels available for any system that you have registered with ULN, see Section 2.4.1, “ULN Channel Subscription Management”.

2.1.3 About Software Errata

Oracle releases important changes to the Oracle Linux and Oracle VM software as individual package updates, known as errata. These package updates are made available for download on ULN before they are gathered into a release or distributed through the `_patch` channel.

Errata packages can contain the following:

- Security advisories, which have names prefixed by `ELSA-*` (for Oracle Linux) and `OVMSA-*` (for Oracle VM).
- Bug fix advisories, which have names prefixed by `ELBA-*` and `OVMBA-*`.
- Feature enhancement advisories, which have names prefixed by `ELEA-*` and `OVMEA-*`.

To be notified when new errata packages are released, you can subscribe to the Oracle Linux and Oracle VM errata mailing lists at https://oss.oracle.com/mailman/listinfo/el-errata and https://oss.oracle.com/mailman/listinfo/oraclevm-errata.

If you are logged into ULN, you can also subscribe to these mailing lists by following the **Subscribe to Enterprise Linux Errata mailing list** and **Subscribe to Oracle VM Errata mailing list** links that are provided on the Errata tab.

Oracle publishes a complete list of errata made available on ULN at https://linux.oracle.com/errata. You can also see a published listing of Common Vulnerabilities and Exposures (CVEs) and explore their details and status at https://linux.oracle.com/cve.

2.1.4 Access Requirements For Restrictive Outbound Firewall Policies

For ULN to function correctly, the host system must have outbound access to `linux-update.oracle.com` via port `443`.

If the outbound firewall you have configured does not support adding exceptions for hostnames, you can use the following IP addresses:

138.1.51.46	ULN IP address from 30 October 2020 at 10pm PT onwards
137.254.56.42	ULN IP address until 30 October 2020 at 10pm PT

2.1.5 For More Information About ULN

You can find out more information about ULN at https://linux.oracle.com/.

2.2 CSI Administration

Access to ULN requires at least one valid Customer Support Identifier (CSI). Your CSI is an identifier that is issued to you when you purchase Oracle Support for an Oracle product. You must provide a valid CSI that covers the support entitlement for each system that you register with ULN.

This chapter describes how you are able to manage and administer CSIs against your user accounts and systems from within ULN.

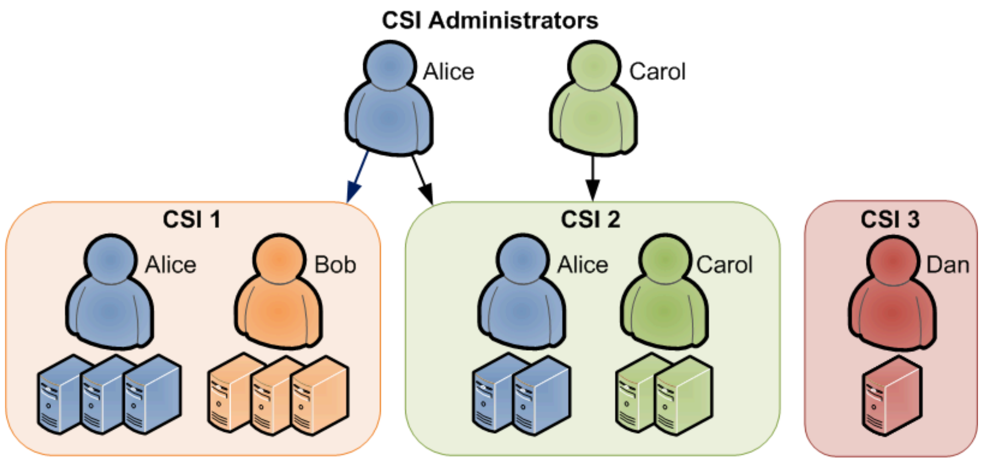
The CSI administration feature of ULN provides a unified view of all of your organization's CSIs and the systems that are registered with those CSIs. To be able to manage the registered systems, you must become an administrator for one or more of your organization's CSIs. To be able to view and change the details of any system that is not registered to your ULN user name, you must become an administrator for the CSI under which that system is registered.

If you are registered as a CSI administrator, you can access the CSI Administration tab while logged in to ULN and perform the following tasks:

- Assign yourself as administrator of a CSI, or assign someone else as administrator of a CSI. See Section 2.2.1, “Becoming a CSI Administrator”.
- List active CSIs, list the servers that are currently registered with an active CSI, and transfer those servers to another user or to another CSI. See Section 2.2.2, “Listing Active CSIs and Transferring Their Registered Servers”.
- List expired CSIs, list the servers that are currently registered with an expired CSI, and transfer those servers to another user or to another CSI. See Section 2.2.3, “Listing Expired CSIs and Transferring Their Registered Servers”.
- Remove yourself or someone else as administrator of a CSI. See Section 2.2.4, “Removing a CSI Administrator”.

Figure 2.1 shows a representative example of an organization with three CSIs, only two of which have CSI administrators.

Figure 2.1 Example of an Organization with three CSIs



CSI 1 has two registered users, Alice and Bob, who each have three systems registered to them.

CSI 2 also has two registered users, Alice and Carol, who each have two systems registered to them.

CSI 3 has one registered user, Dan, who has a single system registered to him.

Alice is registered as an administrator for both CSI 1 and CSI 2. She can view the details of both CSIs, including all systems and users that are registered with those CSIs. She can move systems between CSI 1 and CSI 2, and reassign systems between users in both CSI 1 and CSI 2. She can also assign additional administrators to CSI 1 and CSI 2, or remove administrators from CSI 1 and CSI 2. She cannot see any details for CSI 3.

Carol is registered as an administrator only for CSI 2. She can view the details of that CSI and of all systems and users that are registered with it, including Alice's systems. She can reassign systems between users in CSI 2, but she cannot move systems to the other CSIs. She can assign additional administrators to CSI 2, or remove administrators from CSI 2. She cannot see any details for CSI 1 or CSI 3.

Bob can view only the details of the systems that are registered to him in CSI 1. He cannot see any details for Alice's systems in CSI 1.

Dan is not registered as an administrator for CSI 3. He can view only the details of the system that is registered to him in CSI 3.

Neither Bob nor Dan can perform CSI administration tasks. For example, they cannot move systems between CSIs nor can they reassign systems to other users. However, as CSI 3 does not currently have an administrator, Dan can choose to become its administrator. As CSI 1 already has Alice as its administrator, Bob cannot become an administrator unless Alice grants him that privilege.

For Alice to become an administrator of CSI 3, Dan should register as the administrator of CSI 3 so that he can add Alice as an administrator.

2.2.1 Becoming a CSI Administrator

You can become an administrator of a CSI in one of the following ways:

- When you register with ULN, if no administrator is currently assigned to manage the CSI, you are prompted to click **Confirm** to become the CSI administrator. If you click **Cancel**, you cannot access the CSI administration feature.

- When logged into ULN, if you access the System tab and no administrator is currently assigned to manage one of the CSIs for which you are registered, you are prompted to choose whether to become the CSI administrator.

  To become a CSI administrator:

  1. Click the red link labeled **enter the CSI you would like to be the administrator for in this page**.
  2. On the Add CSI page, verify the CSI and click **Confirm**.

  Note

  On the Systems page, the CSIs of all systems that have no assigned administrator are also shown in red.

- If you are already an administrator of a CSI, you can add yourself as administrator of another CSI provided that you have registered either a server or your ULN user name with the other CSI.

  To assign yourself as administrator of an additional CSI:

  1. Log in to ULN and select the CSI Administration tab.
  2. On the Managed CSIs page, click **Add CSI**.
  3. On the Assign Administrator page, enter the CSI, and click **Add**.
  4. If there are existing administrators, the page lists these administrators and prompts you to click **Confirm** to confirm your request. Each administrator is sent an email to inform them that you have added yourself as an administrator of the CSI.

- An administrator for a CSI can add you as an administrator for the same CSI.

  To assign another administrator to a CSI:

  1. Log in to ULN as administrator of the CSI, and select the CSI Administration tab.
  2. On the Managed CSIs page, click **List Administrators**.
  3. On the CSI Administrators page, click **Assign Administrator**.
  4. On the Assign Administrator page in the Select New Administrator list, click the **+** icon that is next to the user name of the user that you want to add as an administrator. Their user name is added to the **Administrator** box.
  5. If you administer more than one CSI, select the CSI that the user will administer from the **CSI** drop down list.
  6. Click **Assign Administrator**.

  Note

  If you want to become the administrator of a CSI, but the person to whom it is registered is no longer with your organization, contact an Oracle support representative to request that you be made the administrator for the CSI.

2.2.2 Listing Active CSIs and Transferring Their Registered Servers

To list details of the active CSIs for which you are the administrator:

1. Log in to ULN as administrator of the CSI, and select the CSI Administration tab.
2. On the Managed CSIs page in the Select Managed CSI Services pane, select the **Active** link. The Managed Active CSI Services pane displays the service details for each active CSI that you administer.
3. Click the **View # Server(s)** link to display the details of the servers that are registered to an active CSI.
4. On the Registered Servers page, you can transfer one or more systems to another user or to another CSI that you administer.

   Note

   If you transfer a system to another user, at least one of the following conditions must be true:

   - His or her user name must be registered to this CSI.
   - One or more of the servers, for which they are the owner, must be registered to this CSI.
   - He or she must be an administrator of at least one CSI for which you are also an administrator.

   To transfer systems to another user:

   a. Select the **Transfer System** check boxes for the systems that you want to transfer.

   b. Click **Transfer Selected Systems to Another Owner**.

   c. On the Transfer Registered System(s) - Owner page in the Transfer To column, click the red arrow icon that is next to the user name of the user to whom you want to transfer ownership.

   d. On the Confirm Transfer Profile - Owner page, click **Apply Changes** to confirm the transfer to the new owner.

   To transfer systems to another CSI:

   a. Select the **Transfer System** check boxes for the systems that you want to transfer.

   b. Click **Transfer Selected Systems to Another CSI**.

   c. On the Transfer Registered System(s) - CSI page in the Transfer To column, click the red arrow icon that is next to the CSI to which you want to transfer the systems.

   d. On the Confirm Transfer Profile - CSI page, click **Apply Changes** to confirm the transfer to the new CSI.

2.2.3 Listing Expired CSIs and Transferring Their Registered Servers

To list details of the expired CSIs for which you are the administrator:

1. Log in to ULN as administrator of the CSI, and select the CSI Administration tab.
2. On the Managed CSIs page in the Select Managed CSI Services pane, select the **Expired** link. The Managed Expired CSI Services pane displays the service details for each expired CSI that you administer.

3. Click the **View # Server(s)** link to display the details of the servers that are registered to an expired CSI.

4. On the Registered Servers page, you can transfer one or more systems to another user or to another CSI that you administer.

   Note

   If you transfer a system to another user, at least one of the following conditions must be true:

   - His or her user name must be registered to this CSI.
   - One or more of the servers, for which they are the owner, must be registered to this CSI.
   - He or she must be an administrator of at least one CSI for which you are also an administrator.

   To transfer systems to another user:

   a. Select the **Transfer System** check boxes for the systems that you want to transfer.

   b. Click **Transfer Selected Systems to Another Owner**.

   c. On the Transfer Registered System(s) - Owner page in the Transfer To column, click the red arrow icon that is next to the user name of the user to whom you want to transfer ownership.

   d. On the Confirm Transfer Profile - Owner page, click **Apply Changes** to confirm the transfer to the new owner.

   To transfer systems to another CSI:

   a. Select the **Transfer System** check boxes for the systems that you want to transfer.

   b. Click **Transfer Selected Systems to Another CSI**.

   c. On the Transfer Registered System(s) - CSI page in the Transfer To column, click the red arrow icon that is next to the CSI to which you want to transfer the systems.

   d. On the Confirm Transfer Profile - CSI page, click **Apply Changes** to confirm the transfer to the new CSI.

2.2.4 Removing a CSI Administrator

To remove an administrator who is registered for a CSI:

1. Log in to ULN and select the CSI Administration tab.

2. On the Managed CSIs page, click **List Administrators**.

3. On the CSI Administrators page in the Delete? column, click the trash can icon that is next to the user name of the user that you want to remove as administrator for the CSI specified in the same row.

4. When prompted to confirm that you want to revoke administration privileges for the CSI from that user, click **OK**.

2.3 ULN Registration

This chapter describes ULN registration options. Typically, when you register a system with ULN, you provide your Oracle Account credentials as part of the registration process. This action automatically links the system to your own user name and creates a ULN profile for your user. It is also possible to register with ULN without registering a system. Using the approach to register with ULN enables your Oracle Account credentials to be associated with a ULN profile prior to adding any systems to your account. Both approaches are described.

2.3.1 Registering as a ULN User

When you register a system with ULN, your Oracle Account user name is also registered as your ULN user name. If you want to use ULN without first registering a system, you can register as a ULN user provided that you have a valid customer support identifier (CSI) for Oracle Linux support or Oracle VM support. To purchase Oracle Linux or Oracle VM support, go to the online Oracle Linux Store or contact your sales representative.

To register as a ULN user:

1. In a browser, go to https://linux.oracle.com/register.
2. If you do not have an Oracle Account, click **Create New Account** and follow the onscreen instructions to create one.

   If you already have an Oracle Account, click **Sign On**.
3. Log in using your Oracle Account user name and password.
4. On the Create New ULN User page, enter your CSI and click **Create New User**.

Note

If no administrator is currently assigned to manage the CSI, you are prompted to click **Confirm** to become the CSI administrator. If you click **Cancel**, you cannot access the CSI administration feature. See Section 2.2, “CSI Administration”.

If your user name already exists on the system, you are prompted to proceed to ULN by clicking the link **Unbreakable Linux Network**. If you enter a different CSI from your existing CSIs, your user name is associated with the new CSI in addition to your existing CSIs.

2.3.2 Registering an Oracle Linux System With ULN

To register an Oracle Linux system with ULN, follow these steps:

1. Run the `uln_register` command.

   ```
   uln_register
   ```

   If this command is run in a shell outside of a graphical desktop environment, you may be prompted for the `root` password if it is not run by the `root` user. The Registration wizard loads a text user interface menu, where you are able to enter registration information and select options.

   Alternatively, if you use the GNOME graphical user desktop, select **Activities**, and then search for 'ULN Registration'. Click the **ULN Registration** shortcut icon. Note that this application requires system administrator level privileges, so you are prompted for your password to run the application in a desktop environment. This application must be run by a user who is registered as a system administrator on the system.

2. When prompted, enter your ULN user name, password, and customer support identifier (CSI).
3. Enter a name for the system that will enable you to identify it on ULN, then choose whether to upload hardware and software profile data that allows ULN to select the appropriate packages for the system.
4. If you have an Oracle Linux Premier Support account, you can choose to configure a system that is running a supported kernel to receive kernel updates from Oracle Ksplice.

The `dnf-plugin-spacewalk` and `rhn-client-tools` packages are installed and enabled and your system is subscribed to the appropriate software channels.

If you use a proxy server for Internet access, see Section 2.4.3, “Configuring the Use of a Proxy Server”.

For information about registering to use Ksplice, see *Oracle® Linux: Ksplice User's Guide*.

2.4 ULN System Management

You can manage systems that have been registered with ULN by using the ULN web interface at https://linux.oracle.com. These tools enable you to manage system information, such as the associated CSI for a system, the channels that a system is subscribed to, and to browse for available updates and errata.

Some command-line tools are also provided to help with performing certain tasks, such as channel subscription management directly from the shell on the system itself.

System updates and package installation from ULN are handled by using the `dnf` command directly on the system that is registered with ULN.

Some system configuration may be applied directly to the system, for example, to configure proxy settings or to disable ULN updates for a particular package.

This chapter describes the configuration steps and procedures for performing these tasks on a system that is registered with ULN.

2.4.1 ULN Channel Subscription Management

You can configure the channels that a system is subscribed to through the ULN Web interface. The following tasks describe channel subscription in more detail.

2.4.1.1 Managing ULN Channel Subscription by Using the ULN Web Interface

If you have registered your system with ULN, you can subscribe the system to the channels that are available for the level of support that is associated with the CSI.

To subscribe your system to ULN channels:

1. Log in to https://linux.oracle.com with your ULN user name and password.
2. On the Systems tab, click the link named for the system in the list of registered machines.
3. On the System Details page, click **Manage Subscriptions**.
4. On the System Summary page, select channels from the list of available or subscribed channels and click the arrows to move the channels between the lists.

5. When you have finished selecting channels, click **Save Subscriptions**.

Note that you can view a complete listing of all available channels, for all operating systems and all architectures, by clicking on the Channels tab when you are logged into https://linux.oracle.com. You can use the **Release** and **Architecture** drop-down selection boxes to limit the listing to a particular operating system release and architecture.

2.4.2 Modifying System Details

If you have registered your system with ULN, you can modify the details that ULN records for the system.

To update the details for your system:

1. Log in to https://linux.oracle.com with your ULN user name and password.
2. On the Systems tab, click the link named for the system in the list of registered machines.
3. On the System Details page, click **Edit**.
4. On the Edit System Properties page, you can change the name that is associated with your system, register it as a local yum server for your site, or change the CSI with which it is registered.

Note

You cannot change the CSI of a system unless it is registered to your user name.

5. When you have finished making changes, click **Apply Changes**.

2.4.3 Configuring the Use of a Proxy Server

If your organization uses a proxy server as an intermediary for Internet access, specify the `proxy` setting in the `/etc/dnf/dnf.conf` file, as shown in the following example:

```
proxy=http://proxysvr.example.com:3128
```

If the proxy server requires authentication, additionally specify the `proxy_username`, and `proxy_password` settings.

```
proxy=http://proxysvr.example.com:3128
proxy_username=user
proxy_password=password
```

If you use the DNF Spacewalk plugin (`dnf-plugin-spacewalk` and the `rhn-client-tools`) to access the ULN, specify the `enableProxy` and `httpProxy` settings in `/etc/sysconfig/rhn/up2date` as shown in this example.

```
enableProxy=1
httpProxy=http://proxysvr.example.com:3128
```

If the proxy server requires authentication, additionally specify the `enableProxyAuth`, `proxyUser`, and `proxyPassword` settings.

```
enableProxy=1
httpProxy=http://proxysvr.example.com:3128
enableProxyAuth=1
proxyUser=user
```

```
proxyPassword=password
```

Caution

All dnf users require read access to /etc/dnf/dnf.conf or /etc/sysconfig/rhn/up2date. If these files must be world-readable, do not use a proxy password that is the same as any user's login password, and especially not root's password.

2.4.4 Updating a System by Using DNF

DNF integration with ULN makes it possible to run most dnf commands on the system after registering it with ULN and configuring the channels to which the system is subscribed. You can use the dnf install and dnf upgrade commands to handle general package installation or updates.

To update a system to use the latest packages that are available on ULN, run:

```
sudo dnf upgrade
```

The ULN integration with dnf enables you to run commands like dnf repolist to obtain a listing of the ULN channels to which the system is subscribed. You can search for packages and obtain package information in the same way as you would if you were using dnf to access the Oracle Linux yum server.

2.4.5 Disabling ULN Package Updates

You might need to disable package updates from ULN. For example, if you deleted your system from ULN, you would edit the /etc/dnf/plugins/spacewalk.conf file and change the value of enabled flag from 1 to 0 in the [main] section, as shown in the following example:

```
[main]
enabled = 0
gpgcheck = 1
timeout = 120
```

To disable updates for particular packages, add an exclude statement to the [main] section of the /etc/dnf/dnf.conf file. For example, to exclude updates for VirtualBox and kernel:

```
exclude=VirtualBox* kernel*
```

Note

Excluding certain packages from being updated can cause dependency errors for other packages. Your system could also become vulnerable to security-related issues if you do not install the latest updates.

2.4.6 Browsing Available Errata for a System

You can download a comma-separated values (CSV) report file of the errata that are available for a specific system registered on ULN and download any available errata RPMs, individually. You can also browse all of the available advisories that are available on ULN and download the errata RPMs for the supported combinations of the software release and the system architecture.

To download a CSV report or the errata RPMs for a specific system:

1. Log in to https://linux.oracle.com with your ULN user name and password.

2. On the Systems tab, click the link named for the system in the list of registered machines.

The System Details page lists the available errata for the system in the Available Errata table, which might be split over several pages.

3. To download the CSV report file, click the link **Download All Available Errata for this System**.

4. To see more detail about an advisory and to download the RPMs:

   a. Click the link for the advisory.

   b. On the System Errata Detail page for an advisory, you can download the RPMs for the affected releases and system architectures.

   Note that updating the system by using the `dnf upgrade` command directly on the affected system, downloads these RPMs and updates the system with all available errata updates.

To browse all available advisories and download errata RPMs:

1. Log in to https://linux.oracle.com with your ULN user name and password.

2. Select the Errata tab.

   The Errata page displays a table of the available errata for all releases that are available on ULN.

3. On the Errata page, you can perform the following actions on the displayed errata:

   - To sort the table of available errata, click the title of the **Type**, **Severity**, **Advisory**, **Systems Affected**, or **Release Date** column. Click the title again to reverse the order of sorting.

     Note

     The **Systems Affected** column shows how many of your systems are potentially affected by an advisory.

   - To display or hide advisories of different types, select or deselect the **Bug**, **Enhancement**, and **Security** check boxes and click **Go**.

   - To display only advisories for a certain release of Oracle Linux or Oracle VM, select that release from the **Release** drop-down list and click **Go**.

   - To search within the table, enter a string in the **Search** field and click **Go**.

4. To see more detail about an advisory and to download the RPMs:

   a. Click the link for the advisory.

   b. On the Errata Detail page for an advisory, you can download the RPMs for the supported releases and system architectures. The **Superseded By Advisory** column displays a link to the most recent advisory (if any) that replaces the advisory you are browsing.

2.4.7 Removing a System From ULN

To remove a system that is registered with ULN:

1. Log in to https://linux.oracle.com with your ULN user name and password.

2. On the Systems tab, click the link named for the system in the list of registered machines.

3. On the System Details page, click **Delete**.

Note

You cannot delete a system unless it is registered to your user name.

4. When prompted to confirm the deletion, click **OK**.

2.5 Creating and Using a Local ULN Mirror

You can configure a local server to mirror the ULN channels within your network. This approach reduces the overhead that is associated with registering and managing systems within ULN, while still provisioning systems with all of the available software and updates that are available on ULN. Systems that are not able to connect to the Internet, either directly or by using a proxy, can also use this approach to keep up to date with the latest software.

This approach requires that the server is registered with ULN, has the available disk space to host the mirrored channels and is subscribed to the channels that it hosts. Systems that use the ULN mirror are dependent on the synchronization of packages on the local server with the most recent updates provided by Oracle through ULN. If the ULN mirror falls out of date, systems within your network might not be able to install critical security updates.

This chapter provides information on the requirements to host a ULN mirror and the procedure to set up and configure the local server for this purpose. Instructions are provided for configuring client systems to access and use the local server to obtain updates.

Note

If you are considering mirroring ULN channels on a local server, you should also investigate Oracle Linux Manager that is based on the Spacewalk open source software. Oracle Linux Manager provides tools to help with system maintenance, installation and package management, including tools to easily mirror ULN channels either from an intuitive web interface, or from a command line tool. For more information, see the *Oracle® Linux Manager & Spacewalk for Oracle® Linux Documentation*.

2.5.1 Prerequisites for the Local ULN Mirror

The system that you want to set up as a local ULN mirror must meet the following criteria:

- Must be registered with ULN. See Section 2.3, “ULN Registration”.
- Must have at least 6 GB of memory to create the yum metadata.
- Must have enough disk space to store copies of the packages that it hosts. Consider the following when calculating disk space:
  - Disk space requirements depend on the channels to which you subscribe. In turn, the channels depend on the number of clients to be serviced, including their platforms, operating systems, and other specific packages that each client might be using and which would require updates.
  - Disk space that is used for a mirror is only consumed and is never released. Thus, disk requirements are not static and can increase over time.
  - Packages within the channels are also updated regularly and will further affect the storage requirements on the local yum server.

For guidance in estimating the disk size requirements for your specific mirror setup, run the following command, which displays information about each ULN channel to which the system is subscribed:

```
sudo dnf repoinfo
```

To display information only for a specific channel, provide the repository ID in the command, for example:

```
sudo dnf repoinfo ol8_x86_64_baseos_latest
```

```
This system is receiving updates from Unbreakable Linux Network or Spacewalk.
Last metadata expiration check: 0:23:13 ago on Wed 24 Feb 2021 02:52:53 PM GMT.
Repo-id            : ol8_x86_64_baseos_latest
Repo-name          : Oracle Linux 8 BaseOS Latest  (x86_64)
Repo-status        : enabled
Repo-updated       : Tue 23 Feb 2021 07:19:55 PM GMT
Repo-pkgs          : 8,339
Repo-available-pkgs: 8,333
Repo-size          : 19 G
Repo-baseurl       : https://linux-update.oracle.com/XMLRPC/GET-REQ/ol8_x86_64_baseos_latest
Repo-expire        : 172,800 second(s) (last: Wed 24 Feb 2021 02:52:33 PM GMT)
Total packages: 8,339
```

Because repositories are dynamic and grow over time, always plan to allocate substantially greater disk space than what `Repo-size` specifies. The more resources you can provide for disk space, the more efficient the local server can mirror ULN channels.

2.5.2 Setting up a Local ULN Mirror

To set up an Oracle Linux 8 system as a local ULN mirror:

1. Enable the system as a yum server.

   This step disables system specific logic that is applied when a system attempts to subscribe to channels that are not compatible with its architecture or platform. A system that is enabled as a yum server would be able to subscribe to channels for alternate architectures or operating system versions to provide updates to clients with those matching architectures and operating systems.

   Using the ULN web interface:

   a. On your browser, log in at https://linux.oracle.com with the proper credentials.

   b. On the Systems tab, click the link named for the system designated to be a ULN mirror.

   c. On the System Details page, click **Edit**.

   d. On the Edit System Properties page, select the **Yum Server** check box.

   e. Click **Apply Changes**.

   Using the uln-channel command

   a. At the shell of the system designated to be a ULN mirror, run:

   ```
   sudo uln-channel --enable-yum-server
   ```

   b. If prompted, specify the ULN user name and password.

   c. To list other options you can use with the command, type:

```
sudo uln-channel -h
```

2. Subscribe the system to the channels that you intend to mirror.

   Using the ULN web interface:

   a. On the System Details page of the designated ULN mirror, click **Manage Subscriptions**.

   b. On the System Summary page, select channels from the list of available or subscribed channels and click the arrows to move the channels between the lists.

   Note

   If you have an Oracle Linux Premier Support account and you want the yum server to host Ksplice packages for local Ksplice Offline clients, subscribe to the Ksplice for Oracle Linux channels for the architectures and Oracle Linux releases that you want to support.

   c. When you have finished selecting channels, click **Save Subscriptions**.

   Using the uln-channel command

   a. At the shell of the system designated to be a ULN mirror, run:

   ```
   sudo uln-channel -a -c channel [-c channel …]
   ```

   Specify the `-c channel` option multiple times as needed to subscribe to all the ULN channels that you want the server to host.

   b. If prompted, specify the ULN user name and password.

   c. (Optional) To verify that the subscription completed successfully, type:

   ```
   sudo uln-channel -l
   ```

   d. To list other options you can use with the command, type:

   ```
   sudo uln-channel -h
   ```

3. Configure web services on the server.

   a. Install the Apache HTTP server.

   ```
   sudo dnf install httpd
   ```

   b. Create a base directory for the repositories, for example `/var/www/html/yum`.

   ```
   sudo mkdir -p /var/www/html/yum
   ```

   Note

   You can create the yum depository anywhere. However, the yum repository owner must have read and write permissions on that location.

   c. If you created a base directory for the yum repository that is not under `/var/www/html`, for example `/var/yum`, create a symbolic link in `/var/www/html` that points to the repository, for example:

   ```
   sudo ln -s /var/yum /var/www/html/yum
   ```

Further, if SELinux is enabled in enforcing mode, do the following additional steps:

i. Define the default file type of the repository root directory hierarchy as `httpd_sys_content_t`.

```
sudo /usr/sbin/semanage fcontext -a -t httpd_sys_content_t "/var/yum(/.*)?"
```

ii. Apply the file type to the entire repository.

```
sudo /sbin/restorecon -R -v /var/yum
```

d. Edit the HTTP server configuration file, `/etc/httpd/conf/httpd.conf`, as follows:

i. Specify the resolvable domain name or the IP address of the server in the argument to `ServerName`.

```
ServerName ULN-mirror:80
```

ii. Verify that in the `<Directory "/var/www/html">` section, the setting of the `Options` directive specifies `Indexes` and `FollowSymLinks`, for example:

```
Options Indexes FollowSymLinks
```

The setting enables you to browse the directory hierarchy.

e. Start the HTTP server and configure it to start after a reboot.

```
sudo systemctl start httpd
```

```
sudo systemctl enable httpd
```

f. If you enabled a firewall on the system, configure it to allow incoming HTTP connection requests on TCP port 80.

```
sudo firewall-cmd --add-service=http
```

```
sudo firewall-cmd --permanent --add-service=http
```

2.5.3 Updating the Mirrored Repositories on the Local Yum Server

Before updating the repositories on the yum server, create a location for the mirrored repositories, for example:

```
sudo mkdir /repos
```

To update the repositories, use the `reposync` command as follows:

```
sudo dnf reposync --delete --download-metadata -p /repos
```

--delete	This option ensures that any package that is removed upstream is also removed from the local mirror. Using this option is highly recommended.
--download-metadata	This option ensures that all repository metadata are included in the synchronization.

Note

If you are running the command for the first time, the process might take a long while to complete.

For more options that you can use with the command, type:

```
sudo dnf reposync --help
```

2.5.4 Configuring DNF on the Local Yum Server

This configuration is particularly important if the yum server serves multiple clients on varying platforms and operating systems. In this case, the server must subscribe to the channels that are required by those clients. However, the server itself must be updated only with those pacakges that are appropriate for itself. To control the server's own channel and repository subscriptions and avoid updating the server with conflicting packages, you need to configure the server as a client of itself.

1. Use the following command to list the channels that the Oracle Linux 8 yum server is mirroring from ULN:

   ```
   sudo dnf repolist
   ```

   In this example, the server also mirrors the Oracle Linux 7 channels from ULN to be able to service its Oracle Linux 7 clients on the x86_64 platform.

2. Edit `/etc/dnf/plugins/spacewalk.conf` by adding the following stanza for each incompatible channel:

   ```
   [repo_id]
   enabled=0
   ```

   These stanzas disable those incompatible channels and prevent their packages from being installed on the server.

   For example, the server's `/etc/dnf/plugins/spacewalk.conf` would have the following stanzas:

   ```
   [ol7_addons]
   enabled=0

   [ol7_x86_64_latest]
   enabled=0

   [ol7_x86_64_UEKR3_latest]
   enabled=0
   ```

   Note

   If you subsequently subscribe the system to any additional incompatible channels on ULN, you must also disable those channels in `/etc/dnf/plugins/spacewalk.conf`.

3. Configure the server as a yum client as described in Section 2.5.5, “Configuring Client Access to the Local Yum Server” .

2.5.5 Configuring Client Access to the Local Yum Server

The following procedure configures client systems to receive yum updates from the local yum server. The procedure also applies to the local yum server that needs to become a client of itself.

1. Import the GPG key:

   ```
   sudo rpm --import /usr/share/rhn/RPM-GPG-KEY
   ```

2. Disable any existing yum repositories configured in the `/etc/yum.repos.d` directory.

Choose from the following methods:

- Edit each `/etc/yum.repos.d/*.repo` file to specify an `enabled=0` setting for each entry in the file.
- Perform a global disable operation by using `yum-config-manager` or `dnf config-manager`, depending on the client's operating system. For example, on an Oracle Linux 7 client, you would type:

  ```
  sudo yum-config-manager --disable \*
  ```

- Remove the `.repo` extension from the filenames to cause yum operations to ignore these files.

  ```
  sudo cd /etc/yum.repos.d
  ```

  ```
  sudo for i in *.repo; do mv $i $i.disabled; done
  ```

3. Add repository entries from the local yum server.

   You can create a new `/etc/yum.repos.d/local-yum.repo` file and populate it with repository entries that are applicable to the client. The entries must have an `enabled=1` setting, as shown in the following example for an Oracle Linux 7 client:

   ```
   [local_ol7_latest]
   name=Oracle Linux $releasever Latest ($basearch)
   baseurl=http://local_uln_mirror/path-to-mirrored-repo
   gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
   gpgcheck=1
   enabled=1

   [local_ol7_UEKR6_latest]
   name=Unbreakable Enterprise Kernel Release 6 for Oracle Linux $releasever - $basearch - latest
   baseurl=http://local_uln_mirror/path-to-mirrored-repo
   gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
   gpgcheck=1
   enabled=1

   [local_ol7_addons]
   name=Oracle Linux $releasever - $basearch - addons
   baseurl=http://local_uln_mirror/path-to-mirrored-repo
   gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
   gpgcheck=1
   enabled=1
   ```

   For *`local_uln_mirror`*, you can specify either the local yum server's resolvable host name or its IP address.

   Tip

   To distinguish the local repositories from the ULN repositories, prefix the names of their entries with a string such as `local_`.

   Note that the correct GPG key file must exist at the path that is specified for the `gpgkey` parameter. You can download the GPG keys used to sign all of the Oracle Linux release packages from the Oracle Linux yum server. See https://yum.oracle.com/faq.html#a10 for more information.

4. To test the configuration:

   a. Clear the yum metadata cache:

```
sudo yum|dnf clean metadata
```

b. Use `yum|dnf repolist` to verify that the relevant repositories are listed for the client.

   If the client cannot connect to the local yum server, check that the firewall settings on the local yum server allow incoming TCP connections to the HTTP port , which is typically port 80.

5. You can now run `yum|dnf update` on the client to obtain new updates from the local yum server.

2.6 The Unbreakable Linux Network API

This appendix describes the XML-RPC methods that the API provides for access to the Unbreakable Linux Network (ULN).

This API is based on XML-RPC, which enables applications to perform remote operations by encoding the procedure calls in XML and transmitting them over HTTP. For more information about XML-RPC, see http://www.xmlrpc.com/.

The API is accessed at the server entry point URL at https://linux-update.oracle.com/XMLRPC.

The following method namespaces are available:

`auth`	Contains methods for authenticating with ULN. See Section 2.6.1, “Authentication Methods”.
`channel`	Contains methods for listing software channels on ULN. See Section 2.6.2, “Channel Methods”.
`channel.software`	Contains methods for querying packages available within different channels on ULN. See Section 2.6.3, “Channel Software Methods”.
`errata`	Contains methods for interacting with errata on ULN. See Section 2.6.4, “Errata Methods”.
`packages`	Contains methods for querying package information for specified packages on ULN. See Section 2.6.5, “Packages Methods”.
`system`	Contains methods for managing systems registered with ULN. See Section 2.6.6, “System Methods”.

2.6.1 Authentication Methods

Authentication methods are provided in the `auth` namespace. The following methods are provided for authenticating with ULN:

- Section 2.6.1.1, “`auth.login`”
- Section 2.6.1.2, “`auth.logout`”

2.6.1.1 `auth.login`

The `login` method logs in to ULN using a specified user name and password.

The input parameters are:

Input Parameters

`string username`	The Oracle Account user name to use for the session. For example: `myuser@example.com`

`string password`	The password to use for the session. For example: `secret`

Return Parameters

`string sessionKey`	The session key for the session. All other methods use the session key for the duration of the session. For example: `JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc`

2.6.1.2 `auth.logout`

The `logout` method logs out of the ULN session specified by the session key.

Input Parameters

`string sessionKey`	The session key of the session to be terminated. For example: `JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc`

Return Parameters

`int`	The method returns an `int` error code, which indicates whether the session terminated correctly. A value of `1` indicates a successful return.

2.6.2 Channel Methods

Channel methods are available in the `channel` namespace. The following method is provided for listing software channels that are available on ULN:

- Section 2.6.2.1, “`channel.listSoftwareChannels`”

2.6.2.1 `channel.listSoftwareChannels`

The `listSoftwareChannels` method returns a list of software channels that are available to a session on ULN.

Input Parameters

`string sessionKey`	The session key for the session. For example: `JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc`

Return Parameters

`array`	An array of channels with:		
	`struct (channel)`	A structure containing the following strings:	
		`string channel_arch`	The channel architecture. For example: `x86_64`
		`string channel_end_of_life`	The channel end of life. Currently unused

	on ULN.
`string channel_label`	The channel label. For example: `ol7_x86_`
`string channel_name`	The channel name. For example: `Oracle Linux 7 Latest (x86_64)`
`string channel_parent_label`	The channel parent label. Currently unused on ULN.

2.6.3 Channel Software Methods

Channel software methods are available in the `channel.software` namespace. The following methods can by used to query the packages that are available to a session from a channel on ULN.

- Section 2.6.3.1, "`channel.software.getDetails`"
- Section 2.6.3.2, "`channel.software.listAllPackages`"
- Section 2.6.3.3, "`channel.software.listErrata`"
- Section 2.6.3.4, "`channel.software.listLatestPackages`"

2.6.3.1 `channel.software.getDetails`

The `getDetails` method returns the details of the given channel.

Input Parameters

`string sessionKey`	The session key for the session. For example: `JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc`
`string channelLabel`	The channel label for the channel that you wish to query. For example: `ol7_x86_64_latest`

Return Parameters

`string channel_arch_name`	The channel architecture name. For example: `x86_64`

`string channel_description`	The channel description. For example: `All packages released for Oracle Linux 7 (x86_64) including the latest errata packages. (x86_64)`		
`string channel_summary`	The channel summary, usually the same as the channel name. For example: `Oracle Linux 7 Latest (x86_64)`		
`struct metadata_urls`	A dictionary or associative array of metadata locations and checksum information, including the URLs to download channel metadata.		
	`struct filelists`	`string checksum_type`	The hashing algorithm used to generate the checksum. For example: `sha`
		`string checksum`	The checksum for the filelists metadata file. For example: `abc4ef3d6e6`
		`string file_name`	The file name for the filelists metadata at the channel location. For example: `repodata/filelists.x`
		`string url`	The URL where the filelists metadata

		can be accessed. For example: `https:// uln.orac XMLRPC/ GET- REQ/ ol7_x86_ repodata filelist`
`struct group`	This structure is returned optionally if this information is available.	
	`string checksum_type`	The hashing algorithm used to generate the checksum. For example: `sha`
	`string checksum`	The checksum for the group metadata file. For example: `90acbe68`
	`string file_name`	The file name for the group metadata at the channel location. For example:

		`repodata/ comps.xml`
	`string url`	The URL where the group metadata can be accessed. For example: `https:// uln.oracle. XMLRPC/ GET- REQ/ ol7_x86_64_ repodata/ comps.xml`
`struct other`	`string checksum_type`	The hashing algorithm used to generate the checksum. For example: `sha`
	`string checksum`	The checksum for the other metadata file. For example: `20f6b193cd9`
	`string file_name`	The file name for the other metadata at

		the channel location. For example: `repodata other.xm`
	`string url`	The URL where the other metadata can be accessed. For example: `https:// uln.orac XMLRPC/ GET- REQ/ ol7_x86_ repodata other.xm`
`struct primary`	`string checksum_type`	The hashing algorithm used to generate the checksum. For example: `sha`
	`string checksum`	The checksum for the primary metadata file. For example: `3992e1e7`
	`string file_name`	The file name

		for the primary metadata at the channel location. For example: `repodata/primary.xml`
	`string url`	The URL where the primary metadata can be accessed. For example: `https://uln.oracle. XMLRPC/GET-REQ/ol7_x86_64_ repodata/primary.xml`
`struct repomd`	`string file_name`	The file name for the repomd metadata at the channel location. For example: `repodata/repomd.xml`
	`string url`	The URL where the repomd

		metadata can be accessed. For example: `https://uln.orac XMLRPC/GET-REQ/ol7_x86_ repodata repomd.x`
`struct updateinfo`	This structure is returned optionally if this information is available.	
	`string checksum_type`	The hashing algorithm used to generate the checksum. For example: `sha`
	`string checksum`	The checksum for the updateinfo metadata file. For example: `6d11ecbc`
	`string file_name`	The file name for the updateinfo metadata at the channel location. For example:

			repodata/ updateinfo.
		string url	The URL where the updateinfo metadata can be accessed. For example: https:// uln.oracle. XMLRPC/ GET- REQ/ ol7_x86_64_ repodata/ updateinfo.

2.6.3.2 channel.software.listAllPackages

The listAllPackages method returns a list of all packages that are available from a channel, including packages that are not the latest.

Input Parameters

string sessionKey	The session key for the session. For example: JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc.
string channelLabel	The channel label for the channel that you wish to query.

Return Parameters

array	An array of all packages:		
	struct (package)	A structure containing the following strings:	
		string package_arch_label	The package architecture label. For example: noarch
		string package_epoch	The package epoch value, if specified. The

	epoch value can help RPM determine package version ordering if the versioning does not make sense or does not follow sequentiall For example: `1`
`string package_id`	The package ID within the ULN infrastructu For example: `11776733`
`string package_last_modified`	The date and timestamp for when a package was last modified. For example: `2018-09- 19:31:13`
`string package_name`	The name

	of the package. For example: `selinux-policy-mls`
`string package_release`	The package release information. For example: `192.0.6.el7`
`string package_version`	The package version number. For example: `3.13.1`

2.6.3.3 `channel.software.listErrata`

The `listErrata` method returns a list of all errata that are associated with a channel.

Input Parameters

`string sessionKey`	The session key for the session. For example: `JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc`.
`string channelLabel`	The channel label for the channel that you wish to query. For example: `ol7_x86_64_latest`

Return Parameters

`array`	An array of all errata associated with the channel label:		
	`struct (errata)`	A structure containing the following strings:	
		`string errata_advisory_type`	The errata advisory type. For example: `Bug Fix Advisory`
		`string errata_advisory`	The errata

	advisory label. For example: `ELBA-201`
`string errata_issue_date`	The date the errata was issued. For example: `2018-10- 00:00:00`
`string errata_last_modified_date`	The date the errata was last modified . For example: `2018-10- 00:00:00`
`string errata_synopsis`	A brief synopsis of the errata. For example: `glibc bug fix update`
`string errata_update_date`	The errata update date. For example: `2018-10- 00:00:00`

2.6.3.4 `channel.software.listLatestPackages`

The `listLatestPackages` method returns a list of the latest packages that are available from a channel.

Input Parameters

`string sessionKey`	The session key for the session. For example: `JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc`.
`string channelLabel`	The channel label for the channel that you wish to query. For example: `ol7_x86_64_latest`

Return Parameters

`array`	An array of latest packages:		
	`struct (package)`	A structure containing the following strings:	
		`string package_arch_label`	The package architecture label. For example: `noarch`
		`string package_epoch`	The package epoch value, if specified. The epoch value can help RPM determine package version ordering if the versioning does not make sense or does not follow sequentially. For example: `1`

`string package_id`	The package ID within the ULN infrastructu For example: `11776733`
`string package_name`	The name of the package. For example: `selinux-policy-mls`
`string package_release`	The package release information For example: `192.0.6.`
`string package_version`	The package version number. For example: `3.13.1`

2.6.4 Errata Methods

Errata methods are available in the `channel` namespace. The following methods are provided for interacting with errata that are available on ULN:

- Section 2.6.4.1, “`errata.applicableToChannels`”
- Section 2.6.4.2, “`errata.getDetails`”
- Section 2.6.4.3, “`errata.listCves`”
- Section 2.6.4.4, “`errata.listPackages`”

2.6.4.1 `errata.applicableToChannels`

The `applicableToChannels` method returns a list of all channels to which the specified erratum applies. .

Input Parameters

`string sessionKey`	The session key for the session. For example: `JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc`.
`string advisoryName`	The name of the erratum (for example, `ELSA-2013-0269`).

Return Parameters

`array`	An array of channels:		
	`struct (channel)`	A structure containing the following strings:	
		`string channel_id`	The identifier for a channel in the ULN infrastructure. For example: `1844`
		`string channel_label`	The label for the channel. For example: `ol7_x86_64_`
		`string channel_name`	The full name for the channel. For example: `Oracle Linux 7 Latest (x86_64)`
		`string parent_channel_label`	The parent channel label. Not currently

used on ULN.

2.6.4.2 `errata.getDetails`

The `getDetails` method returns detailed information for the specified erratum. Note that the method only fills in the errata_severity field for security errata.

Input Parameters

`string sessionKey`	The session key for the session. For example: `JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc`
`string advisoryName`	The name of the erratum. For example: `ELSA-2013-0269`

Return Parameters

`array`	An array of detailed information associated with the erratum:		
	`struct (erratum)`	A structure containing the following strings:	
		`string errata_description`	The detailed description of the erratum. For example: `[0:1.2.1 Add missing connecti hostname check against X.509 certific name \n- Resolves CVE-2012`
		`string errata_issue_date`	The date the erratum was issued. For example: `2/19/13`
		`string errata_last_modified_date`	The date

	the erratum was last modified: For example: `2013-02-19 00:00:00`
`string errata_notes`	Notes associated with the erratum. Usually empty.
`string errata_references`	References of the erratum. Usually empty.
`string errata_severity`	The severity level set for the erratum . For example: `Moderate`
`string errata_synopsis`	A brief synopsis of the erratum. For example: `axis security update`
`string errata_topic`	The topic for the erratum.

	Usually empty.
`string errata_type`	The type for the erratum. For example: `Security Advisory`
`string errata_update_date`	The errata update date. For example: `2/19/13`

2.6.4.3 `errata.listCves`

The `listCves` method returns a list of Common Vulnerabilities and Exposures (CVE) IDs that are applicable to the specified erratum ID.

Input Parameters

`string sessionKey`	The session key for the session. For example: `JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc`
`string advisoryName`	The name of the erratum. For example: `ELSA-2018-2942`

Return Parameters

`array`	An array of CVE IDs. If no matching CVE IDs are found, the array is empty.:	
	`string cve_name`	The CVE ID associated with the erratum ID. For example: `CVE-2018-3136`

2.6.4.4 `errata.listPackages`

The `listPackage` method returns a list of all packages applicable to the specified erratum ID.

Input Parameters

`string sessionKey`	The session key for the session. For example: `JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc`
`string advisoryName`	The name of the erratum. For example: `ELSA-2018-2942`

Return Parameters

`array`	An array of packages:

`struct (package)`	A structure containing the following strings:		
	`array download_urls`	An array of URLs where the package can be downloaded from.	
		`string url`	
	`array providing_channels`	An array listing channels providing this package.	
		`string labe`	
	`string package_arch_label`	The package architecture label.	

	For example: `i686`
`string package_build_date`	The date and timestamp for when the package was built. For example: `2018-10- 16:39:10`
`string package_build_host`	For example: `x86- ol7- builder-`
`string package_cookie`	The package cookie value. Usually empty.
`string package_description`	The full description of the package. For example: `The OpenJDK demos.`
`string package_epoch`	The package epoch value, if specified. The epoch value can

	help RPM determine package version ordering if the versioning does not make sense or does not follow sequentially. For example: `1`
`string package_file`	The package filename. For example: `java-1.8.0-openjdk-demo-1.8.0.`
`string package_id`	For example: `11807834`
`string package_last_modified_date`	The date and timestamp for when the package was last modified. For example: `2018-10-17 16:39:10`
`string package_license`	The license or

	licenses that a package is released under. For example: `ASL 1.1 and ASL 2.0 and BSD and BSD with advertis and GPL + and GPLv2 and GPLv2 with exceptio and IJG and LGPLv2+ and MIT and MPLv2.0 and Public Domain and W3C and zlib`
`string package_md5sum`	The package md5sum value. For example: `1508de7b`

`string package_name`	The package name. For example: `java-1.8.0-openjdk-demo`
`string package_payload_size`	The package payload size in bytes. For example: `4412184`
`string package_release`	The package release value. For example: `0.el7_5`
`string package_size`	The package size in bytes. For example: `4293131`
`string package_summary`	A summary of the contents of the package. For example: `OpenJDK Demos`
`string package_vendor`	The package vendor name. For

	example: `Oracle America`
`string package_version`	The package version. For example: `1.8.0.19`
`struct package_checksums`	A structure, listing package checksum values by type: `string m`

2.6.5 Packages Methods

Packages methods are available in the `packages` namespace. These methods are used for extracting information about the packages that are available to a session on the ULN. The following methods are provided for interacting with packages that are available on ULN:

- Section 2.6.5.1, "`packages.getDetails`"
- Section 2.6.5.2, "`packages.listProvidingErrata`"

2.6.5.1 `packages.getDetails`

The `getDetails` method returns detailed information about the specified package.

Input Parameters

`string sessionKey`	The session key for the session. For example: `JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc`
`int pid`	The package identifier that should be queried, specified as an integer. For example: `11807834`

Return Parameters

`array`	An array of channels with:

`struct (package)`	A structure containing the following strings:	
	`array download_urls`	An array of URLs where the package can be downloaded from. `string url`
	`array providing_channels`	An array listing channels providing this package. `string labe`
	`string package_arch_label`	The package architecture label.

	For example: `i686`
`string package_build_date`	The date and timestamp for when the package was built. For example: `2018-10-16:39:10`
`string package_build_host`	The host where the package was built. For example: `x86-ol7-builder-`
`string package_cookie`	The package cookie value. Usually empty.
`string package_description`	The full description of the package. For example: `The OpenJDK demos.`
`string package_epoch`	The package epoch

	value, if specified. The epoch value can help RPM determine package version ordering if the versioning does not make sense or does not follow sequentially. For example: `1`
`string package_file`	The package filename. For example: `java-1.8.0-openjdk-demo-1.8.0.`
`string package_id`	For example: `11807834`
`string package_last_modified_date`	The date and timestamp for when the package was last modified. For

	example: `2018-10- 16:39:10`
`string package_license`	The license or licenses that a package is released under. For example: `ASL 1.1 and ASL 2.0 and BSD and BSD with advertis and GPL + and GPLv2 and GPLv2 with exceptio and IJG and LGPLv2+ and MIT and MPLv2.0 and Public Domain and W3C and zlib`
`string package_md5sum`	The package

		md5sum value. For example: `1508de7bafe`
	`string package_name`	The package name. For example: `java-1.8.0-openjdk-demo`
	`string package_payload_size`	The package payload size in bytes. For example: `4412184`
	`string package_release`	The package release value. For example: `0.el7_5`
	`string package_size`	The package size in bytes. For example: `4293131`
	`string package_summary`	A summary of the contents of the package. For example: `OpenJDK Demos`

`string package_vendor`	The package vendor name. For example: `Oracle America`
`string package_version`	The package version. For example: `1.8.0.19`
`struct package_checksums`	A structure, listing package checksum values by type: `string m`

2.6.5.2 `packages.listProvidingErrata`

The `listProvidingErrata` method returns a list of the errata that are associated with a package.

Input Parameters

`string sessionKey`	The session key for the session. For example: `JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc`
`int pid`	The package identifier that should be queried, specified as an integer. For example: `11807834`

Return Parameters

`array`	An array of all errata associated with the package:	
	`struct (errata)`	A structure containing the following strings:

`string errata_advisory_type`	The errata advisory type. For example: `Security Advisory`
`string errata_advisory`	The errata advisory label. For example: `ELSA-2018-2`
`string errata_issue_date`	The date the errata was issued. For example: `2018-10-17 00:00:00`
`string errata_last_modified_date`	The date the errata was last modified . For example: `2018-10-17 00:00:00`
`string errata_synopsis`	A brief synopsis of the errata. For example: `java-1.8.0-openjdk security update`

`string errata_update_date`	The errata update date. For example: `2018-10-00:00:00`

2.6.6 System Methods

System methods are available in the `system` namespace. These methods are used for managing systems that are registered on ULN. The following methods are available:

- Section 2.6.6.1, “`system.deleteSystems`”

2.6.6.1 `system.deleteSystems`

The `deleteSystems` method removes a system from ULN, given its system ID.

Input Parameters

`string sessionKey`	The session key for the session. For example: `JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc`.
`string serverId`	The system identifier that should be removed. This needs to be the id value within ULN. For example: `330213`

Return Parameters

`int`	The method returns an `int` error code, which indicates whether the system was deleted or not. A value of `0` indicates that the system was successfully removed from ULN.

Appendix A Application Stream Module Life Cycle

Table of Contents

While the core operating system packages in the BaseOS repository for Oracle Linux 8 retain a standard Oracle Linux support life cycle, the separate AppStream packages have their own major version releases and shorter lifespans from 2 to 5 years.

Support for the AppStream packages is limited to package installation assistance only. Additional support for AppStream modules might be provided if references to these modules and their use are included in other official Oracle Linux documentation from Oracle. Critical security errata and select high-impact critical bug fixes are provided in the newer versions of AppStream packages.

The following table lists AppStream packages currently in Oracle Linux 8. As best practice, you should upgrade to the latest release of these packages as possible. Oracle continues to provide support for modules in the AppStream channel until the specified retirement date, as listed in the table.

Module	Stream	Release Date	Retirement Date	Release
`Apache Subversion`	`subversion 1.14`	May 2021	May 2024	8.4
`authd`	`authd 1.4.4`	July 2019	May 2021	8.0.0
`container-tools`	`container-tools 1.0`	July 2019	May 2021	8.0.0
	`container-tools 2.0`	May 2020	May 2022	8.2.0
	`container-tools 3.0`	May 2021	April 2023	8.4
`dotnet`	`dotnet 2.1`	July 2019	August 2021	8.0.0
	`dotnet 3.0`	November 2019	March 2020	8.1.0
	`dotnet 3.1`	February 2020	December 2022	8.1.1
`freeradius`	`freeradius 3.0`	July 2019	May 2024	8.0.0
`gcc-toolset`	`gcc-toolset 9`	November 2019	November 2021	8.1.0
	`gcc-toolset 10`	November 2020	November 2022	8.3
`git`	`git 2`	July 2019	July 2029	8.0.0
`glusterFS`	`glusterFS 6`	March 2020	May 2022	8.1.0
`glusterFS`	`glusterFS 8`	November 2021	To be defined	8.5.0
`httpd`	`httpd 2.4`	July 2019	July 2029	8.0.0
`Identity Management`	`Identity Management DL1`	July 2019	May 2024	8.0.0
`mailman`	`mailman 2.1`	July 2019	June 2024	8.0.0
`mariadb`	`mariadb 10.3`	July 2019	July 2029	8.0.0
	`mariadb 10.5`	May 2021	May 2026	8.4

Module	Stream	Release Date	Retirement Date	Release
`maven`	`maven 3.5`	July 2019	May 2022	8.0.0
	`maven 3.6`	May 2020	April 2023	8.2.0
`mercurial`	`mercurial 4.8`	July 2019	May 2022	8.0.0
`mysql`	`mysql 8`	July 2019	April 2023	8.0.0
`nginx`	`nginx 1.14`	July 2019	May 2021	8.0.0
	`nginx 1.16`	November 2019	November 2021	8.1.0
	`nginx 1.18`	November 2020	November 2022	8.3
`nodejs`	`nodejs 10`	July 2019	April 2021	8.0.0
	`nodejs 12`	November 2019	April 2022	8.1.0
	`nodejs 14`	November 2020	April 2023	8.3
`openjdk`	`openjdk 1.8.0`	July 2019	May 2026	8.0.0
	`openjdk 11`	July 2019	October 2024	8.0.0
`perl`	`perl 5.24`	July 2019	May 2021	8.0.0
	`perl 5.30`	November 2020	November 2023	8.3
`php`	`php 7.2`	July 2019	May 2021	8.0.0
	`php 7.3`	November 2019	November 2021	8.1.0
	`php 7.4`	November 2020	July 2029	8.3
`postgresql`	`postgresql 9.6`	July 2019	November 2021	8.0.0
	`postgresql 10`	July 2019	May 2024	8.0.0
	`postgresql 12`	February 2020	July 2029	8.1.1
	`postgresql 13`	May 2021	May 2026	8.4
`python`	`python 2.7`	July 2019	June 2024	8.0.0
	`python 3.8`	May 2020	May 2023	8.2.0
	`python 3.9`	May 2021	May 2024	8.4
`redis`	`redis 5`	July 2019	May 2022	8.0.0
	`redis 6`	May 2021	May 2024	8.4
`ruby`	`ruby 2.5`	July 2019	July 2029	8.0.0
	`ruby 2.6`	November 2019	March 2022	8.1.0
	`ruby 2.7`	November 2020	March 2023	8.3
`scala`	`scala 2.1`	July 2019	May 2022	8.0.0
`swig`	`swig 3`	July 2019	May 2022	8.0.0
`varnish`	`varnish 6`	July 2019	May 2022	8.0.0

A.1 Rolling Application Streams Release Life Cycle

Rolling application streams are supported for the full life of the Oracle Linux 8 release. New versions of the streams replace existing versions in update releases. Rolling streams are only used when having new versions of the stream is very important. Users of rolling streams should understand when and how the streams are updated and be prepared for newer versions.

Rolling Application Stream	Release Date	Version
`LLVM`	May 2021	11.0.0
`Go`	May 2021	1.15.7
`Rust`	May 2021	1.49