Oracle Linux

Managing Software in Oracle Linux

F19386-43
November 2023

Oracle Linux Managing Software in Oracle Linux,

F19386-43

Contents

Preface

Conventions vi
Documentation Accessibility vi
Access to Oracle Support for Accessibility vi
Diversity and Inclusion vii

1 How Oracle Distributes Software Packages

Distributing Packages Through the Oracle Linux Yum Server 1-1
- Available Oracle Linux Yum Servers 1-1
- Available Yum Repositories 1-1
- Securing the Distribution of Oracle Linux Packages 1-2

Distributing Packages Through the Unbreakable Linux Network 1-3
- Comparing ULN and Yum Servers 1-3
- Accessing ULN 1-3
- About ULN Channels 1-4
  - Main ULN Channels for Oracle Linux 9 1-4
  - Main ULN Channels for Oracle Linux 8 1-5

About the DNF Utility 1-7

2 Configuring a System to Use Oracle Linux Yum Server

Configuring the Global DNF Configuration Settings 2-1
Configuring a System to Use a Proxy With a Yum Server 2-2
Configuring Access to the Oracle Linux Yum Server Through a Firewall 2-3
Subscribing to Different Yum Repositories 2-3
- Editing Yum Repository Configuration Files 2-3
- Configure Compute Instances Access to Regional Yum Server Repositories 2-4
- Using the DNF config-manager Plugin 2-5
- How to Recover the Base Yum Repository Configuration 2-5

3 Configuring a System to Use ULN

How to Register a System With ULN 3-1
Configuring a System to Use a Proxy for ULN 3-2
How to Manage a System's Channel Subscriptions 3-3
How to Change System Details in ULN 3-3
How to Remove a System From ULN 3-4

4 Installing Software on Oracle Linux

Using DNF Groups 4-2
Using DNF Modules and Application Streams 4-2
- About Modular Dependencies and Stream Changes 4-4
- How to Enable Modules 4-5
- How to Remove Installed Modules 4-6
- How to Switch Module Streams 4-7

5 Updating Software on Oracle Linux

Updating Software Automatically 5-1
Disabling Updates for Particular Packages 5-2
Tracking Security Updates and Errata Releases 5-2
- Using DNF to See Security Updates 5-3
- How to Use ULN to Manage System-Specific Errata 5-5
- How to Use ULN to Browse Available Errata 5-5

Planning for Controlled Updates in a Production Environment 5-6

6 Using Software Distribution Mirrors

Prerequisites for the Local Distribution Mirror 6-1
How to Set Up a Distribution Mirror 6-2
Setting Up a Local Yum Mirror 6-4
- How to Configure the Local Yum Server 6-4
- How to Use rsync to Mirror the Oracle Linux Yum Server 6-8
- How to Mirror Repositories From an ISO 6-8

Setting Up a Local ULN Mirror 6-9
- How to Configure the Local ULN Mirror 6-9
- How to Localize Subscriptions for the ULN Mirror 6-11

How to Configure Client Access to the Local Mirror 6-12

7 DNF Command References

8 Comparing Yum Version 3 With DNF

A Application Stream Module Life Cycle

B Managing ULN Users

Registering to Use ULN B-3
Becoming a CSI Administrator B-3
Listing Active CSIs and Transfer Their Registered Servers B-4
Listing Expired CSIs and Transferring Their Registered Servers B-5
Removing a CSI Administrator B-6

C About the Unbreakable Linux Network API

Authentication Methods C-1
auth.login C-1
auth.logout C-2
Channel Methods C-2
channel.listSoftwareChannels C-2
Channel Software Methods C-3
channel.software.getDetails C-3
channel.software.listAllPackages C-7
channel.software.listErrata C-8
channel.software.listLatestPackages C-8
Errata Methods C-9
errata.applicableToChannels C-9
errata.getDetails C-10
errata.listCves C-11
errata.listPackages C-11
Packages Methods C-14
packages.getDetails C-14
packages.listProvidingErrata C-16
System Methods C-17
system.deleteSystems C-17

Preface

Oracle Linux: Managing Software on Oracle Linux provides information about how to install, upgrade, and manage software on Oracle Linux 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 isn't available in the repositories that are provided by the Oracle Linux yum server.

If you're using Oracle Linux 7, see Oracle Linux 7: Managing Software and Oracle Linux: Unbreakable Linux Network User's Guide for Oracle Linux 6 and Oracle Linux 7.

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/.

For information about the accessibility of the Oracle Help Center, see the Oracle Accessibility Conformance Report at https://www.oracle.com/corporate/accessibility/templates/t2-11535.html.

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.

1

How Oracle Distributes Software Packages

Oracle uses two mechanisms to distribute software packages:

- Oracle Linux Yum Server
- Unbreakable Linux Network (ULN)

Depending on the infrastructure and the support agreement with Oracle, you can use either of these software distribution mechanisms with the Oracle Linux systems you're running. You can also create software distribution mirrors to provision software to a broader infrastructure.

Distributing Packages Through the Oracle Linux Yum Server

Instead of using the installation media, you can access the Oracle Linux yum server to install Oracle Linux packages, including bug fixes, security fixes, and enhancements. Oracle logically organizes software packages on the yum server into different repositories based on package purpose, support status, or dependencies.

Available Oracle Linux Yum Servers

Two Oracle Linux yum sources for package distribution are available:

Public Yum Server
The primary Oracle Linux yum server is publicly available at https://yum.oracle.com/ where you can obtain software packages for free.
The repositories in the public yum server are replicates of a subset of ULN channels. Channels that contain software, such as Ksplice, that are only licensed for use by Oracle Linux Support customers are unavailable in the server. For more information, see Available Yum Repositories.

Oracle Cloud Infrastructure Yum Servers
Unlike the publicly available yum server, Oracle replicates *all* ULN channels to the Oracle Cloud Infrastructure yum servers. Thus, compute instances have access to software directly without requiring ULN registration. Access to specific ULN content depends on the support contract that you have for an Oracle Cloud Infrastructure account.
To enable access to restricted content through the regional yum servers, ensure that you have installed the appropriate `release-el8` packages and have enabled the repositories to which you require access.

Available Yum Repositories

A yum repository is a directory of packages that are typically available on a web server or an ISO image. The directory also includes metadata in a `repodata` subdirectory. The metadata is updated each time a package changes within the repository directory.

You can configure any client system to use a yum repository by creating a yum repository configuration entry. To install software from the repository, you use either the `yum` or `dnf` command to install software from the repository.

In Oracle Linux, yum repository names map to equivalent ULN channel names, but excluding the platform architecture. For example, the ULN channel `ol8_x86_64_baseos_latest` is `ol8_baseos_latest` on the Oracle Linux yum server. Yum repository names don't include the platform architecture because the URL to the repository already identifies the architecture. Therefore, when accessing the yum server, the system is automatically connected to the appropriate architecture's repositories.

Core OS repositories are the minimum required repositories for an Oracle Linux system to function. These repositories are enabled immediately after installation and must remain enabled through the life cycle of an Oracle Linux system.

On Oracle Linux 9 systems, the core OS repositories are `ol9_baseos_latest` and `ol9_appstream`.

On Oracle Linux 8 systems, the core OS repositories are `ol8_baseos_latest` and `ol8_appstream`.

For a complete list of available repositories on the Oracle Linux yum server, go to https://yum.oracle.com and under the *Browse the Repositories* section, click the link that corresponds to the system's Oracle Linux version.

For additional information, see the Oracle Linux Yum Server Frequently Asked Questions.

Securing the Distribution of Oracle Linux Packages

For access, the public Oracle Linux yum server is configured to use the HTTPS protocol, hence the URL https://yum.oracle.com. The protocol implements a signed SSL certificate that validates the connection with other parties. The communication that's established between the server and other systems is encrypted as a protection against interference when packages are downloaded.

Also, when building packages, Oracle uses Gnu Privacy Guard (GnuPG or GPG) encryption. GPG works through private and public keys. In a network channel where an exchange of files or packages occurs, the recipient uses the public key to authenticate the source and validate the source as trustworthy.

The system's repository files contain parameters that are related to GPG, as shown in the following entry in `/etc/yum.repos.d/oracle-linux-ol8.repo`:

```
[ol8_baseos_latest]
name=Oracle Linux 8 BaseOS Latest ($basearch)
baseurl=https://yum$ociregion.$ocidomain/repo/OracleLinux/OL8/baseos/
latest/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1
...
```

- `gpgkey`: specifies the full path of the key that's provided by the repository maintainer.
- `gpgcheck=1`: the default `1` setting indicates that package installation also automatically includes GPG key verification that ensures that the packages to be

installed are trusted packages. Always ensure that `gpgcheck=1` is the persistent setting.

The public keys that Oracle generates for Oracle Linux packages are available on the Oracle Linux yum server and are included when the packages are installed on the system. The public GPG key is installed automatically when you install the `oraclelinux-release` package.

You can update the public keys by downloading them from the Oracle Linux yum server.

On Oracle Linux 8, run the following commands:

```
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
```

On Oracle Linux 9, run the following commands:

```
sudo wget https://yum.oracle.com/RPM-GPG-KEY-oracle-ol9 -O /etc/pki/rpm-gpg/
RPM-GPG-KEY-oracle
sudo gpg --import --import-options show-only /etc/pki/rpm-gpg/RPM-GPG-KEY-
oracle
```

Distributing Packages Through the Unbreakable Linux Network

The Unbreakable Linux Network (ULN) uses channels to distribute software packages. Each channel contains a logical grouping of packages based on the Oracle Linux version, platform architecture, and package purpose. To access packages, you subscribe to the channels that you require.

Comparing ULN and Yum Servers

Using ULN has advantages over yum. ULN contains access to extra software that's not available through the public Oracle Linux yum server. Most notably, ULN provides access to Oracle Ksplice software channels so that you can automatically update the system kernel without requiring a reboot, along with several other channels for commercially available software from Oracle. Therefore, you can download useful packages that aren't included in the original distribution.

ULN offers software patches, updates, and fixes for Oracle Linux and Oracle VM, and information about `yum`, `dnf`, Ksplice, and support policies. The ULN Alert Notification Tool periodically checks with ULN and alerts you when updates are available.

Accessing ULN

To access ULN, you must be an Oracle Linux Support customer with a valid Customer Supports Identifier (CSI) and a Single Sign-On (SSO) account. Then, you can use the comprehensive resources of ULN at https://linux.oracle.com/. This site provides a web interface where you can review and manage the software channels available to different systems and platforms.

To use `dnf` with ULN, you must individually register each system with ULN and subscribe the system to one or more ULN channels. When you register a system with ULN, the system

automatically chooses the channel that contains the latest version according to the system's architecture and OS release.

About ULN Channels

Channels correspond to the architecture of a system. The Unbreakable Linux Network has more than 100 unique channels. These 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. ULN channels also exist for MySQL, Oracle VM, Oracle Ksplice, OCFS2, RDS, and productivity applications. Other channels might also become available, such as channels for the beta versions of packages, or for specific developer content.

ULN channels are of the following types:

Core
Consists of required channels of a specific Oracle Linux release, including the `*_latest` channel which distributes the latest possible version of any package release. Registered systems are automatically subscribed to appropriate core channels.

> **Caution:**
>
> Unsubscribing from the `_latest` channel can make the system vulnerable to security-related issues. We recommend that you keep the system subscribed to this channel.

Base and Patch
Extra ULN channels that are available for various OS update levels or revisions. You can maintain a system at a specific update by unsubscribing from the `_latest` channel and replacing it with `_base` and `_patch` channels. However, this configuration can leave a system vulnerable to security issues because Oracle stops updating the patch channels after releasing a new update level. Also, software in the `_appstream` channel is always released in line with the latest release. Fixing the system to a particular update level could create dependency issues when Oracle updates the software in the `_appstream` channel.

Not all channels are available for all architectures. Use the ULN web interface to check what channels are available for a specific system architecture. See How to Manage a System's Channel Subscriptions. See also selected channels for Oracle Linux 9 and Oracle Linux 8 and their respective descriptions.

Main ULN Channels for Oracle Linux 9

The following table lists the primary ULN channels for Oracle Linux 9. Additional channels are available. Check the ULN web interface for a complete list.

Channel	Description
ol9_*arch*_baseos_latest	Core channel. Provides all the latest versions of the base operating system packages in the current release of 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.
ol9_*arch*_appstream	Core channel. Provides all the latest versions of the Application Stream user space packages in the current release of 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.
ol9_*arch*_addons	Provides packages released by Oracle in addition to the upstream packages made available in the other channels listed here. These packages are specific to functionality that Oracle provides to improve user experience on Oracle Linux and to provide access to services specific to Oracle.
ol9_*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.
ol9_*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 ol9_*arch*_appstream channel. Support for the codeready_builder packages is limited to package installation assistance only.

Main ULN Channels for Oracle Linux 8

The following table lists the primary ULN channels for Oracle Linux 8. Additional channels are available. Check the ULN web interface for a complete list.

Channel	Description
ol8_*arch*_baseos_latest	Core channel 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	Core channel 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*_un_baseos_base	Provides the base versions of the base operating system packages in the distribution when a particular update level is released. 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. If you are subscribed to the _baseos_latest channel, you do not need to subscribe to this channel.
ol8_*arch*_un_baseos_patch	Provides the patched versions of the base operating system packages in the distribution 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 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. If you are subscribed to the _baseos_latest channel, you do not need to subscribe to a patch 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 specific to functionality that Oracle provides to improve user experience on Oracle Linux and to provide access to services specific to Oracle.

Channel	Description
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. Support for the codeready_builder packages is limited to package installation assistance only.
ol8_*arch*_developer	Provides packages intended for developers to create test and development environments for Oracle Linux 8 and related technologies. Support for the developer packages is limited to package installation assistance only.
ol8_*arch*_developer_EPEL	Provides a mirror of the selected packages that are available on the EPEL (Extra Packages for Enterprise Linux) repository. Support for the EPEL packages is limited to package installation assistance only.

About the DNF Utility

The dnf utility, which is based on Dandified Yum (DNF), is the client software for installing and managing packages on systems running Oracle Linux 8 or later releases. These packages can come from either the Oracle Linux yum server or from ULN. While installing or upgrading packages, dnf also automatically handles package dependencies and requirements.

DNF provides significant improvements in functionality and performance when compared to the traditional yum command. DNF also brings a host of new features, including modular content, and a more stable and documented API. DNF is compatible with Yum v3 for editing or creating configuration files and for managing repositories and packages. You can use the dnf command and all its options in the same manner as how you use the yum command on previous releases of Oracle Linux.

To provide backward compatibility, the `yum` and `dnf` commands are interchangeable. You not only can perform tasks similar to those that you performed in earlier releases of Oracle Linux, but you can also avail of a wider range of new features that are available in `dnf`, such as improved package management and performance. To view syntax differences between `dnf` and legacy `yum` commands, see Comparing Yum Version 3 With DNF.

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

> **Important:**
>
> Oracle Linux packages are built as RPM packages. However, avoid using the `rpm` command for install or update operations unless explicitly instructed to do so by a support representative. In particular, if you do use the `rpm` command, never use the `--force` or `--nodeps` options. Otherwise, you might cause serious system stability issues.

For more information, see the `dnf(8)` manual page and https://dnf.readthedocs.io/en/latest/index.html.

2

Configuring a System to Use Oracle Linux Yum Server

Define global configuration options in the `dnf.conf` file. Then edit, or create `.repo` files to define which repositories to subscribe to.

Configuring the Global DNF Configuration Settings

Edit the main configuration in `/etc/dnf/dnf.conf`. The global definitions for DNF are located under the `[main]` section heading of the DNF configuration file. The following table lists important directive for DNF.

> **Note:**
>
> For backward-compatibility purposes, a symbolic link to `/etc/dnf/dnf.conf` is created at `/etc/yum.conf`. The configuration syntax is the same; although, some configuration options have been deprecated and some new configuration options have been added. See Comparing Yum Version 3 With DNF for a list of the differences between configuration options and syntax.

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

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 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 `dnf` log file.
`obsoletes`	If set to 1, replace obsolete packages during upgrades.
`plugins`	If set to 1, enable plugins that extend the functionality of `dnf`.

Directive	Description
`proxy`	URL of a proxy server including the port number. See Configuring a System to Use a Proxy With a Yum Server
`proxy_password`	Password for authentication with a proxy server.
`proxy_username`	Username for authentication with a proxy server.
`reposdir`	Directories where `dnf` looks for repository files with a `.repo` extension. The default directory is `/etc/yum.repos.d`. See Subscribing to Different Yum Repositories.

Example [main] Configuration

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
```

Configuring a System to Use a Proxy With a Yum Server

If the 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, also specify the `proxy_username`, and `proxy_password` settings.

```
proxy=http://proxysvr.example.com:3128
proxy_username=user
proxy_password=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, don't use a proxy password that's the same as any user's login password, and especially not `root`'s password.

Configuring Access to the Oracle Linux Yum Server Through a Firewall

The Oracle Linux yum server delivers content through a Content Delivery Network (CDN). When you connect to the Oracle Linux yum server, you connect to a node on the CDN that's geographically closer to the system you're using. With CDN, download speeds are faster.

Because the CDN has many nodes that run on different networks around the world, you can not configure any single IP address or network range for an egress firewall rule.

In environments where a strict firewall policy limits outbound connections for systems, we recommend configuring a local yum mirror server within a demilitarized zone (DMZ). See Setting Up a Local Yum Mirror for information on how to configure a local yum mirror.

Subscribing to Different Yum Repositories

Oracle Linux uses modular yum repository configuration files that are made available as release packages that are maintained through yum. Release packages simplify repository management and also ensures that yum repository definitions are kept up-to-date automatically whenever you update the system.

On all Oracle Linux systems, the `oraclelinux-release-el8` package is installed by default. This package contains the core repository configurations to access all the repositories required for an Oracle Linux system to install common OS software packages and the other release packages used to obtain other yum repository configurations.

A list of all available RPM files to manage all the possible yum repository configurations for a release can be obtained by running the following command, where *n* is the Oracle Linux release version of the RPM files, such as `-el8`.

```
dnf list "*release-eln*"
```

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

Editing Yum Repository Configuration Files

DNF uses yum repository configuration files to configure where to install different packages and their dependencies from. By default, repository configuration files are stored in the `/etc/yum.repos.d` directory. You can define another directory location to store repository configurations by setting the `reposdir` directive in the dnf.conf file.

Use the repository directory to define `.repo` files for repositories that you want to make available. A `.repo` file can contain entries for more than one yum repository. To subscribe to a repository, you can edit the `enabled` option to a value of `1` and save the configuration file. The change has immediate effect.

The following table describes the basic directives for a repository. Any other directive that appears in the repository file override the corresponding global definition in the `[main]` section of the DNF configuration file. See the `dnf.conf(5)` manual page for more information.

Directive	Description
`baseurl`	Location of the repository (expressed as a `file://`, `ftp://`, `http://`, or `https://` address). This directive must be specified.
`enabled`	Whether to enable `dnf` to use the repository. Set the value to 1 to enable the repository, or 0 to disable the repository.
`name`	Descriptive name for the repository channel. This directive must be specified.

Example Repository Listing

The following listing shows an example repository section from a `.repo` 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


[el8_appstream]
name=Oracle Linux $releasever Application Stream ($basearch)
baseurl=https://yum.eldistribution.com/repo/EnterpriseLinux/EL8/
appstream/$basearch
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-el
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*.

Configure Compute Instances Access to Regional Yum Server Repositories

Compute instances in Oracle Cloud Infrastructure have access to regional yum servers through the service gateway. The base URL of the repository uses the `$ociregion` variable to define which regional server to use and the `$ocidomain` variable to define the domain where the yum server is located. By using variables,

configuration can remain relatively standard across Oracle Linux deployments but provide access to extra resources available to Oracle Cloud Infrastructure customers.

For example, the base URL to the `ol8_baseos_latest` repository for Oracle Linux 8 is:

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

You can set the `$ociregion` variable by populating content in `/etc/dnf/vars/ociregion`. For example, if `$ociregion` is set to `-phx`, the base URL expands to point to the regional yum server located in Phoenix.

Typically, when you create an instance, this value is set to point to the closest regional yum server on the Oracle Cloud Infrastructure service network. If the `/etc/dnf/vars/ociregion` file doesn't exist, or the file is empty, the base URL points to the publicly accessible Oracle Linux yum server.

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 existing configuration. The most significant of these utilities 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 that's 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 of which the repository is hosted. For example, to create a configuration file in `/etc/repos.d` for an example repository, run the following command:

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

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

```
sudo dnf config-manager --enable myrepo
```

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.

How to Recover the Base Yum Repository Configuration

Perform this task if the system's base repository configuration has been corrupted or otherwise lost.

1. Create a temporary repository configuration file in `/etc/yum.repos.d`.

   ```
   sudo mkdir /etc/yum.repos.d/temp_base.repo
   ```

2. Populate the file with entries corresponding to the system's OS version.
   - For Oracle Linux 9:

     ```
     [ol9_baseos_latest]
     name=Oracle Linux 9 BaseOS Latest ($basearch)
     baseurl=https://yum$ociregion.$ocidomain/repo/OracleLinux/OL9/
     baseos/latest/$basearch/
     gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
     gpgcheck=1
     enabled=1
     ```

   - For Oracle Linux 8:

     ```
     [ol8_baseos_latest]
     name=Oracle Linux 8 BaseOS Latest ($basearch)
     baseurl=https://yum$ociregion.$ocidomain/repo/OracleLinux/OL8/
     baseos/latest/$basearch/
     gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
     gpgcheck=1
     enabled=1
     ```

3. Reinstall the required release packages of the system's OS version to set up the standard yum repository configurations.

   ```
   sudo dnf reinstall oraclelinux-release-ol<release-nbr>
   ```

4. Verify that the recovery is successful.

   ```
   ls /etc/yum.repos.d/oraclelinux-release-ol<release-nbr>
   ```

5. Remove the temporary configuration file.

   ```
   rm /etc/yum.repos.d/temporary_base.repo
   ```

6. Reinstall other required release packages to obtain the correct repository configurations.

   ```
   sudo dnf reinstall <repository>
   ```

7. Enable the repositories that you need.

   ```
   sudo dnf config-manager --enable <repository>
   ```

3
Configuring a System to Use ULN

After you install Oracle Linux on a system, by default the system uses the public Oracle Linux yum server as the source for the system's repository needs. However, you also have the option to configure the system to use ULN, which requires that you register the system with ULN.

To see the differences between the yum server and ULN, see Comparing ULN and Yum Servers.

How to Register a System With ULN

Registering a system with ULN provides an advantage of obtaining access to extra software packages that aren't available in the public Oracle Linux yum server.

To register with ULN, the following requirements must be met:

- You must be an Oracle Linux Support customer with a valid Customer Support Identifier (CSI) and a Single Sign-On (SSO) account.
- The system user account must have system administrator privileges.
- Systems behind a firewall must have outbound access to `linux-update.oracle.com` through port 443. If the outbound firewall doesn't support adding exceptions for hostnames, then use the IP address 138.1.51.46.

This task provides steps to register with ULN by using either the command line or the desktop graphical user interface.

1. Access the ULN registration form.
   - **Using the command line.**
   a. Verify that the `rhn-setup` package is installed.

      ```
      sudo dnf list rhn-setup
      ```

      If the package isn't installed, type:

      ```
      sudo dnf install rhn-setup
      ```

   b. Access the ULN registration form.

      ```
      uln_register
      ```

   c. Provide the credentials as prompted.

      The Set Up Software Updates window is displayed.

   - **Using the desktop graphical user interface**
   a. On the desktop, select **Activities**, then search for ULN Registration.
   b. Click the **ULN Registration** shortcut icon.

c. Provide the credentials as prompted.

   The Set Up Software Updates window is displayed.

2. Click **Next**.
3. Provide the ULN username, password, and customer support identifier (CSI).
4. Enter a ULN name for the system.
5. Choose whether to upload hardware and software profile data that enable ULN to select the appropriate packages for the system.
6. If you have an Oracle Linux Support account and the system is running an appropriate kernel, configure a system to receive kernel updates from Oracle Ksplice.

   This step installs and enables the `dnf-plugin-spacewalk` and `rhn-client-tools` packages and subscribes the system to the appropriate software channels.

   If you use a proxy server for Internet access, see Configuring a System to Use a Proxy for ULN.

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

Configuring a System to Use a Proxy for ULN

If you use the yum plugin (`yum-rhn-plugin`) to access 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, also 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, don't use a proxy password that's the same as any user's login password, and especially not `root`'s password.

How to Manage a System's Channel Subscriptions

Subscribing a system to ULN channels causes the system to automatically receive package updates when these become available on those channels.

Ensure that the system is registered with ULN. See How to Register a System With ULN.

You use the ULN web console to subscribe to channels or removing channel subscriptions for a system.

1. Log in to https://linux.oracle.com with the appropriate ULN user name and password.
2. (Optional) View the available channels to which you can subscribe the system.
   a. Click the **Channels** tab.
   b. Use the **Release** and **Architecture** drop-downs to limit the listing to a particular OS release and architecture.
3. Manage the system's subscription information.
   a. Click the **Systems** tab and from the list of registered machines, select the system whose subscriptions you want to manage.
   b. On the System Details page, click **Manage Subscriptions**.
   c. 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.

      Moving channels between the available list and the subscribed list either adds or removes a channel subscription.

      > **Caution:**
      >
      > Unsubscribing from the `_latest` channel can make the system vulnerable to security-related issues. We recommend that you always keep the system subscribed to the `_latest` channel.

4. After you have finished selecting channels, click **Save Subscriptions**.

How to Change System Details in ULN

Change system information updates and keep the system's registration information current in ULN.

The system must be registered to the username with which you connect to ULN. Otherwise, you can not complete this task.

To complete this task, use the ULN web console.

1. Sign in to https://linux.oracle.com with the appropriate ULN username 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's associated with the system, register the system as a local yum server for other systems, or change the CSI with which the system is registered.

5. After completing the changes, click **Apply Changes**.

How to Remove a System From ULN

Unregistering a system from ULN removes the system from automatically receiving package updates.

The system must be registered to the username with which you connect to ULN. Otherwise, you can not complete this task.

Remove a system from ULN if you prefer to have control over how and when to update the system's packages. You use the ULN web console to complete these steps.

1. Log in to https://linux.oracle.com with the appropriate ULN username 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**.
4. To confirm the deletion, click **OK**.
5. Disable the system's automatic updates from ULN.

   Edit the `/etc/dnf/plugins/spacewalk.conf` file by changing the value of `enabled` option to `0` as shown:

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

6. Subscribe the system to the appropriate yum repositories on the Oracle Linux yum server or an appropriate mirror to receive software updates for bug fixes and security patches.

4

Installing Software on Oracle Linux

Regardless of whether you use ULN or an Oracle Linux yum server, software packages are installed on a system by using standard `dnf` commands and depend on the system having the appropriate ULN channel subscriptions or yum repositories enabled.

Use the `dnf install` command to install a package and any of its dependencies:

```
sudo dnf install package_name
```

> **Note:**
>
> `dnf` makes no distinction between installing and upgrading a kernel package. Both `dnf install` and `dnf upgrade` always download the latest kernel available for the system's OS.

The system notifies you of any extra packages that might be installed and prompts you to confirm whether to go ahead with the installation:

```
Last metadata expiration check: 0:03:03 ago on Mon 11 Sep 2023 11:48:58 AM
GMT.
Dependencies resolved.
================================================================================
==========================
 Package              Architecture          Version
Repository              Size
================================================================================
==========================
Installing:
 dnf-automatic        noarch                4.14.0-5.0.1.el9_2
ol9_baseos_latest        53 k

Transaction Summary
================================================================================
==========================
Install  1 Package

Total download size: 53 k
Installed size: 52 k
Is this ok [y/N]:



Last metadata expiration check: 0:03:03 ago on Mon 11 Sep 2023 11:48:58 AM
GMT.
Dependencies resolved.
```

```
================================================================================
================================
 Package              Architecture
Version                       Repository              Size
================================================================================
================================
Installing:
 dnf-automatic        noarch
4.14.0-5.0.1.el9_2           el9_baseos_latest        53 k

Transaction Summary
================================================================================
================================
Install  1 Package

Total download size: 53 k
Installed size: 52 k
Is this ok [y/N]:
```

You can bypass the confirmation check in a `dnf install` command by using the `-y` option.

For a list of `dnf` commands that are commonly used to manage DNF packages and repositories, see DNF Command References.

Related Topics

- Subscribing to Different Yum Repositories
- How to Manage a System's Channel Subscriptions
  Subscribing a system to ULN channels causes the system to automatically receive package updates when these become available on those channels.

Using 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 subgroups 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.

When installing a group package, use the following command:

```
sudo dnf group install group-name
```

For a list of `dnf` commands that are commonly used to manage DNF groups, see Table 7-2 of DNF Command References.

Using DNF Modules and Application Streams

DNF introduces the concepts of modules, streams, and profiles for managing different versions of software applications within a single OS release. Modules can be used to group many packages that consist of a single application and its dependencies.

Streams can be used to provide alternative 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 other packages for enhanced functionality.

Modular content is made available separate to core OS packages so that these user-space applications can be installed in various user-space environments, including virtual machines, containers, and the base OS. Modular content for Oracle Linux 8 and Oracle Linux 9 is typically shipped within the Application Stream (AppStream) repository.

For a list of application stream packages in the latest Oracle Linux version, see Oracle Linux: Product Life Cycle Information.

- **Modules**: Are a set of RPM packages that are grouped together and must be installed together. They can contain several streams that consist of several 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 several 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. Often, the stream with the latest version is selected as the default stream and is used when operations don't specify a particular stream or a different stream hasn't been enabled before.

  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, with version 10 being the current default stream.

  > **Note:**
  >
  > We recommend that you use the latest stream for any module that's installed, even though other streams might 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 doesn't prevent you from installing or uninstalling any of the packages that are provided by the module. Furthermore, you can install packages by using the profiles of the same module without any further preparation. Also, a module's package list can contain packages from outside of the module stream. These packages are often from BaseOS or the stream's own dependencies. Note that modules in Application Stream always have a default profile. This default profile is used for installations, when no other profile is explicitly specified.

For example, The `httpd` module that includes the Apache web server includes 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 don't require extra 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's installed, the application behaves exactly as any other native application that you have installed by using DNF.

To manage modules, you use a combination of `dnf` and `dnf module` subcommands. For a list of `dnf` commands that are commonly used with modules, see Table 7-3 of DNF Command References.

About Modular Dependencies and Stream Changes

Typically, packages that provide content depend on other packages and specify the appropriate 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 extra constraints. For example, module streams can declare dependencies on the streams of other modules, independent of the packages that the modules contain and provide. After any package or module operation, the entire dependency tree for all the underlying installed packages must satisfy all the conditions that the packages declare. Likewise, all the module stream dependencies must satisfied.

These extra constraints require that you analyze and understand the implications before performing any package operation. Changing the enabled module streams doesn't 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 other modules.
- Installing a module stream profile or packages from a stream might also require the enabling of streams of other modules and the installation of extra 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 other 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 later; thereby, mirroring the behavior of an unused yum repository.
- Switching the stream that's enabled for a module is the same as resetting the current stream and enabling a new stream.

> **Note:**
>
> Switching an enabled stream doesn't 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, and the installation of packages in a new stream are all tasks that must be performed manually.

- Because of potential upgrade scripts that run during an installation, directly installing a stream of a module, other than one that's already installed by default, isn't 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 are used in Application Stream only when a modular package is provided by a module stream that's 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 enabled stream of the module that provides this package and doesn't upgrade to a version from a different stream.

How to Enable Modules

Selected modules are available by default when you install Oracle Linux. You can install these modules as needed. Note that directly installing a stream of a module, other than one that's installed by default, might cause unexpected issues.

This task uses the example of the PHP module to show you how to enable modules.

As an alternative, you can the `dnf module switch-to` command for this task. See Table 7-3.

1. Check the status of the module.

```
sudo dnf module list php
```

```
Oracle Linux 8 Application Stream (x86_64)
Name            Stream             Profiles
Summary
php             7.2 [d]            common [d], devel, minimal          PHP
scripting language
php             7.3                common [d], devel, minimal          PHP
scripting language
php             7.4                common [d], devel, minimal          PHP
scripting language
php             8.0                common [d], devel, minimal          PHP
scripting language

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

   Each PHP version has 3 corresponding profiles, with the common profile as the default. The `[d]` flag indicates the default version of the profile.

2. List the packages that are installed for the profiles associated with the module stream.

```
sudo dnf module info --profile php:8.0
```

The output displays each profile and its associated packages that are required for the module. For example:

```
Name    : php:8.0:8060020211215065547:0a326c83:x86_64
common  : php-cli
        : php-common
        : php-fpm
        : php-mbstring
        : php-xml
...
```

3. Enable the module stream you want to use.

```
sudo dnf module enable php:8.0 -y
```

4. Optionally, check the status of the selected stream to verify that it has the `e` flag.

```
sudo dnf module list php


Name           Stream
Profiles                              Summary
php            7.2 [d]            common [d], devel,
minimal          PHP scripting language
php            7.3                common [d], devel,
minimal          PHP scripting language
php            7.4                common [d], devel,
minimal          PHP scripting language
php            8.0 [e]            common [d], devel,
minimal          PHP scripting language
```

5. To switch to a preferred module stream, see How to Switch Module Streams.

How to Remove Installed Modules

Removing an installed module removes all the packages and their dependencies that are installed by the profiles of the enabled module stream.

The modules to be removed must have some profiles already installed.

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

This task only removes packages that are listed in profiles of the operational module stream. Packages that aren't listed in any of module stream's profiles remain installed on the system and can be removed manually. At each step, a summary of pending changes is first displayed and you're prompted to confirm the action before proceeding.

1. Remove the module.

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

2. Disable the module stream.

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

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

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

How to Switch Module Streams

By switching module streams, you can obtain a different version of the current module that's being used in the system.

The module stream that you want to switch must already be enabled *and*, at the same time, another stream of the same module must already exist.
Before switching module streams, review the information in About Modular Dependencies and Stream Changes.

Switching to a different module stream effectively causes the content to be either upgraded or downgraded to a version that's different from the current version on the system.
In some steps, a summary of pending changes is first displayed and you're prompted to confirm the action before proceeding.

You can also use the `dnf module switch-to` command for this task. See Table 7-3.

1. Reset the module to enable you to install an alternative stream.

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

2. Install the profiles of a different stream of the module.

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

   You might also need to apply changes to other module streams and packages.

3. Update or downgrade any packages installed from the previous module stream that weren't listed in the profiles installed in the previous step.

   ```
   sudo dnf distro-sync
   ```

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

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

5

Updating Software on Oracle Linux

Regardless of whether you use ULN or an Oracle Linux yum server, software updates are achieved using standard `dnf` commands on the system and depend on the system having the appropriate ULN channel subscriptions or yum repositories enabled.

You can use the `dnf install` and `dnf update` commands to handle general package installation or updates.

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

```
sudo dnf upgrade
```

> **Note:**
>
> The `dnf update` command automatically runs `dnf upgrade`.

Update the system often to ensure that packages have the latest security patches and bug fixes. Consider using automatic updates so that software is correctly maintained on the system.

Note than if you use the `dnf install` command for software that's already installed, the software packages that you specify are also updated to the latest available version.

Updating Software Automatically

The DNF Automatic tool is provided as an extra package that you can use to keep the system automatically updated with the latest security patches and bug fixes. The tool can provide 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 -y 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 other timer units are available and can override the default configuration that's specified in the configuration file. Often, 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 doesn't 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
```

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

> **Important:**
>
> By using Oracle Ksplice, you can keep an Oracle Linux kernel patched and updated all the time, without any need to reboot. For Oracle Linux Support customers, Ksplice is an essential tool to keep the systems safe, secure, and updated. See Oracle Linux: Ksplice User's Guide for more information.

Disabling Updates for Particular Packages

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. A system could also become vulnerable to security-related issues if you don't install the latest updates.

Tracking Security Updates and Errata Releases

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're 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 in to 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.

You can also track updates to Oracle Linux yum server repositories by visiting https://yum.oracle.com/whatsnew.html, where you can see which packages were updated within each repository for the previous six months.

> **Note:**
>
> Oracle doesn't comment on existing security vulnerabilities except through Errata announcements at https://linux.oracle.com/errata. To provide the best security posture to all Oracle customers, Oracle fixes significant security vulnerabilities in severity order. So, the most critical issues are always fixed first. Fixes for security vulnerabilities are produced in the following order:
>
> - Latest code line refers to the code being developed for the next major Oracle release of the product.
> - Next patch set for all non terminal releases

Using DNF to See Security Updates

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

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

```
sudo dnf updateinfo list
```

The output from the command sorts the available errata in order of their IDs and identifies their types, which can be one of the following:

- Security patch (*severity*`/Sec.`)
- Bug fix (`bugfix`)
- Feature enhancement (`enhancement`)

Security patches are also listed according to their severity, which can be `Critical`, `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=Critical
```

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 specific CVE ID, for example:

```
sudo dnf updateinfo list --cve CVE-2022-3545
```

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

```
sudo dnf updateinfo info --cve CVE-ID
```

To update all 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 don't 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` command. For Enterprise Linux Security Advisory (ELSA) patches, use `dnf update --advisory`.

```
sudo dnf update --cve CVE-ID
```

```
sudo dnf update --advisory ELSA-ID
```

> **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.

How to Use ULN to Manage System-Specific Errata

Monitoring available errata in ULN keeps you current on updates that might be needed on registered systems.

You can only manage errata for systems that are registered with ULN.

With this task, you can download a CVS report about errata that affect a specific system. Through the report, you can identify the necessary RPMs to download to update that system.

1. Log in to https://linux.oracle.com with the appropriate ULN username 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. Click **Download All Available Errata for this System**.

   As an alternative, use the `sudo dnf upgrade` command directly on the affected system to download the RPMs and update the system with all available errata updates.
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.

How to Use ULN to Browse Available Errata

Monitoring available errata in ULN keeps you current on updates that might be needed on registered systems.

You can only monitor errata for systems that are registered with ULN.

With this task, you can browse all available errata directly in ULN and then select to download the errata RPMs that registered systems require.

1. Log in to https://linux.oracle.com with the appropriate ULN username 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 systems are potentially affected by an advisory.

- To display or hide advisories of different types, select or clear 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** 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.

Planning for Controlled Updates in a Production Environment

Software and OS updates can pose a problem for complex production environments that have mission critical applications that require minimal downtime. One solution might be to lock an environment to a single tested Oracle Linux release and update level to avoid updating the OS often. However, this approach increases the risk from security vulnerabilities and can make integration testing more difficult.

We recommend that you implement a software update strategy to ensure that the OS and underlying software packages on production systems are often updated in a way that you can manage the risk of application breakages because of software updates.

The following guidelines can help you to implement a software update strategy that's in line with best practice but protects the production systems from unexpected changes.

- **Create a local ULN mirror.**
  One of the challenges associated with rolling out updates on systems is that even if you have tested the updates in an integration and testing environment, if you don't manage the source of the updated packages, changes to packages can occur between the period of integration testing and the moment when you roll the package updates out to the production environment.

  By creating a local ULN mirror, you can control when and how often channels are synchronized to the mirror server. The selection of packages is static between synchronization periods, which gives you an opportunity to test a set of packages and then update the production environment to a known working set.

  By using ULN for the mirror service, you can mirror channels that contain Ksplice updates so that you can take advantage of an offline Ksplice service. With the offline Ksplice, you can use in-memory kernel updates to avoid reboots. At the same time, you can test these updates in an integration environment first, before applying the updates to the production environment.
- **Consider a staged update strategy based on risk and threat mitigation.**

Not all updates are equal. You can time synchronization of ULN Mirror channels depending on requirements. Based on those requirements, you can configure systems to perform different update types on differing schedules. For example, you can work with a strategy similar to the following:

- Schedule Oracle Ksplice updates for the kernel and user space to run at least weekly. Optionally, you can vet these updates within an integration test environment first.
- For security related package updates, follow a monthly maintenance schedule and in line with alerts from security tools or errata notifications. Use the `dnf update --security` command for these types of update.
- Apply at least a quarterly maintenance schedule to run full package updates that use a ULN mirror snapshot. Vet the updates on an integration test environment first before implementing these on production servers.

By performing regular atomic updates it's easier to resolve integration issues as they arise and you better protect an environment from potential security issues. Using an integration test environment and a Yum or ULN mirror is critical to maintaining stability of a platform and protecting it from compromise.

6

Using Software Distribution Mirrors

Distribution mirrors are alternative sources of software packages to repositories on the Oracle Linux yum server or Unbreakable Linux Network. These are selected repositories that you locally replicate from the public server. The local repositories become the package sources for client systems that exist in the local network.

Distribution mirrors are useful in complex infrastructures and are important when developing a controlled update strategy for a mission critical production environment. Distribution mirrors are deployed to provide the following services:

- Provide access to yum repositories or ULN channels for systems that don't have access to a public network.
- Improve software download times and reducing bandwidth overhead for larger infrastructure
- Set up network-based installation infrastructure
- Cater for a snapshot style update strategy where testing can be performed against a controlled software distribution environment before the updates are implemented on production systems.

A server that functions as a software distribution mirror contains yum repositories or ULN channels. The repositories or channels can be made available to client systems in the internal network through various methods such as using local web server, a file transfer server, and so on.

The software distribution mirror must be synchronized with the official Oracle Linux sources. If required, you can control synchronization to occur at strategic intervals so that you can test system updates against a known set of package versions before you roll them out to all the infrastructure.

> **Note:**
>
> If you're considering mirroring ULN channels on a local server, check also Oracle Linux Manager that's based on the Spacewalk open source software. Oracle Linux Manager provides tools to help with system maintenance, installation, and package management. For more information, see Oracle® Linux Manager & Spacewalk for Oracle® Linux Documentation.

Prerequisites for the Local Distribution Mirror

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

- Must have Internet access to connect to the official Oracle Linux sources.
- Has at least 6 GB of memory to create the yum metadata.
- Must be configured to provide access to the mirrored repositories by system clients.

- Has enough disk space to store copies of the packages that it hosts.

  When calculating for the needed disk space, consider the following:

  – Disk space requirements depend on the repositories or channels that you choose to mirror. Other factors are 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's used for a mirror is only consumed and is never released. Thus, disk requirements aren't static and can increase over time.

  – Packages in the repositories or channels are also updated on a regular schedule and further affects the storage requirements on the local yum server.

  For guidance in estimating the disk size requirements, run the following command:

  ```
  sudo dnf repoinfo [repo-ID]
  ```

  Part of the command output includes the size of a specific repository, for example:

  ```
  ...
  Repo-id            : ol8el8_x86_64_baseos_latest
  ...
  Repo-size          : 29 G
  ...

  Repo-id            : ol8el8_addons
  ...
  Repo-size          : 4.8 G
  ...
  ```

  Because repositories are dynamic and grow over time, always plan to allocate substantially greater disk space than what `Repo-size` specifies. Optionally, you can also create a dedicated file system and mount this to the directory that hosts the mirrored repositories.

How to Set Up a Distribution Mirror

Systems can be configured to distribute packages and provide updates to client systems within a local network without the need for clients to access the public servers through the Internet.

You can select any method to provide access to the local repositories in the mirror server. This task uses HTTP as an example.

1. Ensure that the latest version of the `yum-utils` is installed on the system.

   ```
   sudo dnf install -y yum-utils
   ```

2. Install the Apache HTTP server.

   ```
   sudo dnf install -y httpd
   ```

3. Create a base directory for the local repositories, for example:

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

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

4. If you created a dedicated file system for the mirror, them mount that file system to the base directory.

5. If you created a base directory in a different location than `/var/www/html`, create a symbolic link in `/var/www/html` that points to the repository.

   For example, if the base directory is `/var/yum`, type:

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

6. If SELinux is enabled in enforcing mode, do the following steps:

   a. 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/
      repos(/.*)?"
      ```

   b. Apply the file type to the entire repository.

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

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

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

      ```
      ServerName system-mirror:80
      ```

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

      ```
      Options Indexes FollowSymLinks
      ```

      With this setting, you can browse the directory hierarchy.

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

   ```
   sudo systemctl start httpd
   sudo systemctl enable httpd
   ```

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

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

10. Choose how you want the local mirror to function to serve clients in the local network:

- Set up the system as a yum mirror.
- Set up the system as a ULN mirror.

11. Continue to set up the system as a local yum mirror as described in Setting Up a Local Yum Mirror.

Setting Up a Local Yum Mirror

A system that functions as a local yum repository mirrors repositories from the public Oracle Linux yum server.

When Oracle Linux is installed on this system, that system automatically contains the repositories that are required by the system's OS. These repositories are found in the system's `/etc/yum/repos.d` directory. The repositories are defined in different `/etc/yum/repos.d/*.repo` files.

By mirroring these default repositories, the system can function as a local yum server to service clients that have the same OS and platform as the mirror.

However, you might want the local yum mirror to also service clients that use different OS releases for other platforms. In this case, you would need to define other repositories that are required by those clients.

How to Configure the Local Yum Server

Setting up a system to function as a local yum server involves mirroring required repositories from the public Oracle Linux yum server.

The yum mirror must meet the requirements described in Prerequisites for the Local Distribution Mirror. Also, you must have completed the procedure in How to Set Up a Distribution Mirror.

You can mirror any repository available on the Oracle Linux yum server, if you have the definition for the repository configured in `/etc/yum.repos.d`. Mirroring repositories that the system already has available is uncomplicated. However, for other repositories, you might need to be more specific about the which repositories you want to mirror. Moreover, you might need to provide other repository configuration.

1. Mirror all the current system's enabled repositories to the base directory.

   ```
   sudo dnf reposync --delete --download-metadata -p /var/www/html/yum
   ```

   --delete
   Remove from the mirror any package that's removed upstream. Using this option is highly recommended.

   --download-metadata
   Include all repository metadata in the synchronization.

   If you run the command for the first time, the process might take a long while to complete. At the end of the process, the system becomes ready to provide packages to client systems with compatible OS and platforms as the mirror.

2. Set the local mirror to host repositories for heterogeneous clients.

a. Create the required repositories for mixed clients.

Suppose that the server is running the latest Oracle Linux 8 release, but must provide packages for Oracle Linux 9 and Oracle Linux 7 clients. You would do the following:

- Create `/etc/yum.repos.d/9-mirror.repo` with entries similar to the following example:

```
[ol9_baseos_latest]
name=Oracle Linux 9 BaseOS Latest  ($basearch)
baseurl=https://yum$ociregion.$ocidomain/repo/OracleLinux/OL9/
baseos/latest/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=0

[ol9_appstream]
name=Oracle Linux 9 Application Stream Packages ($basearch)
baseurl=https://yum$ociregion.$ocidomain/repo/OracleLinux/OL9/
appstream/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=0


[el9_baseos_latest]
name=Oracle Linux 9 BaseOS Latest  ($basearch)
baseurl=https://yum.example.com/repo/EnterpriseLinux/EL9/baseos/
latest/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=0

[el9_appstream]
name=Oracle Linux 9 Application Stream Packages ($basearch)
baseurl=https://yum.example.com/repo/EnterpriseLinux/EL9/
appstream/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=0
```

- Create `/etc/yum.repos.d/7-mirror.repo` with entries similar to the following example:

```
[ol7_latest]
name=Oracle Linux 7 Latest ($basearch)
baseurl=https://yum$ociregion.$ocidomain/repo/OracleLinux/OL7/
latest/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=0

[ol7_optional_latest]
name=Oracle Linux $releasever Optional Latest ($basearch)
baseurl=https://yum$ociregion.$ocidomain/repo/OracleLinux/OL7/
optional/latest/$basearch/
```

```
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=0

[ol7_addons]
name=Oracle Linux $releasever Add ons ($basearch)
baseurl=https://yum$ociregion.$ocidomain/repo/
OracleLinux/OL7/addons/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=0


[el7_latest]
name=Oracle Linux 7 Latest ($basearch)
baseurl=https://yum.example.com/repo/EnterpriseLinux/EL7/
latest/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=0

[el7_optional_latest]
name=Oracle Linux $releasever Optional Latest ($basearch)
baseurl=https://yum.example.com/repo/EnterpriseLinux/EL7/
optional/latest/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=0

[el7_addons]
name=Oracle Linux $releasever Add ons ($basearch)
baseurl=https://yum.example.com/repo/EnterpriseLinux/EL7/
addons/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=0
```

Full yum configurations for different releases are available at https://yum.oracle.com/mirror/. Navigate to the correct distribution and architecture and download the appropriate `*.repo` files.

> **Important:**
>
> All entries must have `enabled=0` to prevent packages from these repositories to be installed on the local yum mirror itself.

b. Mirror each repository in the `*.repo` files to the base directory.

```
sudo dnf reposync --repoid el9ol9_baseos_latest --delete --download-
metadata -p /var/www/html/yum
...


sudo dnf reposync --repoid el9ol97_latest --delete --download-
metadata -p /var/www/html/yum
...
```

3. Automate the regular client package updates from mirrored repositories through a cron script or systemd timer unit.

   For example, create a file at `/etc/cron.daily/yum-mirror-update` with the following content:

```
#!/bin/bash
# Regularly update yum repos
dnf reposync --delete --download-metadata -p /var/www/html/yum
```

   Ensure that the file is executable.

```
sudo chmod +x /etc/cron.daily/yum-mirror-update
```

   - If the yum mirror services mixed clients, change the script to resemble the following:

```
#!/bin/bash

REPOS=(repo-IDs)

for REPO in ${REPOS[@]}
do
    dnf reposync --repo=$REPO --delete --download-metadata -
p /var/www/html/yum
done
```

     repo-IDs represents a comma separated list of the IDs of repositories that are required by **all** the clients of the mirror. These IDs are contained in corresponding `/etc/yum.repos.d/*.repo` files you created for those clients. In this procedure's example, the *repo-IDs* would represent Oracle Linux 8 repositories for clients that are compatible with the mirror. In addition, you would include `el9_baseos_latest`, `el9_appstream`, `el7_latest`, `el7_optional_latest`,`el7_addonsol9_baseos_latest`, `ol9_appstream`, `ol7_latest`, `ol7_optional_latest`,`ol7_addons`, and so on, for the other clients.

4. Configure clients appropriately to access these repositories.

   See How to Configure Client Access to the Local Mirror.

How to Use rsync to Mirror the Oracle Linux Yum Server

Oracle provides an `rsync` interface to the Oracle Linux yum server repositories at the `yum-rsync.oracle.com` domain that maps directly to the URL structure of the Oracle Linux yum server.

You must fulfill the requirements as described in Prerequisites for the Local Distribution Mirror. Additionally, you must complete the procedure as provided in How to Set Up a Distribution Mirror.

With the `rsync` interface, you can easily mirror the Oracle Linux yum server for broader usage without any requirement for complex system configuration. This approach is helpful for large enterprises that want to mirror entire repository structures for all architectures. The `rsync` interface is an alternative method to running the `reposync` command to synchronize mirrored repositories.

1. Install `rsync` on the system.

   ```
   sudo dnf install -y rsync
   ```

2. Use `rsync` to mirror all the repositories that you intend to mirror.

   For example, to mirror all the Oracle Linux 8 repositories for all architectures, you can recursively mirror everything at the `rsync://yum-rsync.oracle.com/repo/OracleLinux/OL8/` endpoint.

   ```
   rsync -arv rsync://yum-rsync.oracle.com/repo/
   OracleLinux/OL8 /var/www/html/yum/
   ```

   You can mirror a particular repository for a particular architecture by providing a more specific URL. For example, to mirror the current Oracle Linux 9 `baseos` repository for the x86_64 architecture, you would type:

   ```
   mkdir -p /var/www/html/yum/OL9/baseos/latest
   rsync -arv rsync://yum-rsync.oracle.com/repo/OracleLinux/OL9/baseos/
   latest/x86_64 /var/www/html/yum/OL9/baseos/latest/
   ```

How to Mirror Repositories From an ISO

The local yum mirror can be configured to mirror repositories from an ISO image to make them available to clients.

You must fulfill the requirements as described in Prerequisites for the Local Distribution Mirror. You must also complete the procedure as provided in How to Set Up a Distribution Mirror.

This task assumes that you're mirroring the repositories from an Oracle Linux 8 image. It also assumes that to provide access to the mirror, you're using a web server.

1. Mount the ISO image at an appropriate location so you can copy its contents.

   ```
   sudo mount -o loop,ro OL8EL8.iso /mnt
   ```

2. Create a directory to host the repositories from the ISO.

```
sudo mkdir -p /var/www/html/yum/8_ISO
```

3. Copy the repositories from the ISO to the new directory.

```
sudo cp -r /mnt/BaseOS /var/www/html/yum/8_ISO/
sudo cp -r /mnt/AppStream /var/www/html/yum/8_ISO/
```

4. Configure clients appropriately to access these repositories.

   See How to Configure Client Access to the Local Mirror.

Setting Up a Local ULN Mirror

A system that functions as a local ULN server mirrors channels in the Unbreakable Linux Network.

When you register an Oracle Linux system with ULN, that system is automatically subscribed to default channels in ULN, depending on the system's OS release and architecture. As such, the system can become a mirror to service clients that have the same OS and platform as the mirror.

However, you might also want the local ULN mirror to service clients that use different OS releases for other platforms. In this case, you would need to subscribe to any other channels that are required by those clients.

> **Note:**
>
> Mirroring ULN channels is often slower than mirroring yum repositories. Only consider creating a ULN mirror for channels that aren't otherwise available on the Oracle Linux yum server. Where possible, set up mirrors of Oracle Linux yum server repositories instead.

How to Configure the Local ULN Mirror

Setting up the system to be a local ULN mirror involves replicating channels from Unbreakable Linux Network.

The designated ULN mirror must meet the requirements described in Prerequisites for the Local Distribution Mirror. Additionally, you must have completed the followng tasks:

- Set up the system as a distribution mirror.
- Registered the system with ULN.

For each step in this procedure, you can use either the ULN web interface or the `uln-channel` command. To display options that you can use with the `uln-channel` command, type `uln-channel -h`.

1. Enable the system as a yum server.

   As a yum server, the system can subscribe to channels for OS versions and platforms other than the system's own OS and platform.

- **Using the ULN web interface**

a. On a 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. On the system's terminal window, type:

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

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

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 Support account and you want the mirror 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. On the system's terminal window, type:

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

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

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

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

3. Protect the system's own repositories when other mirrored repositories are being updated.

   See How to Localize Subscriptions for the ULN Mirror.

4. Mirror the ULN Channels to the location of the base directory for the mirror, by using the `dnf reposync` command.

   ```
   sudo dnf reposync --delete --download-metadata -p /var/www/html/yum
   ```

   Consider creating a cron script or systemd service and timer to run this command regularly. For example, create a file at `/etc/cron.daily/uln-mirror-update` with the following content:

   ```
   #!/bin/bash
   # Regularly update yum repos
   dnf reposync --delete --download-metadata -p /var/www/html/yum
   ```

   Ensure that the file is executable.

   ```
   sudo chmod +x /etc/cron.daily/uln-mirror-update
   ```

How to Localize Subscriptions for the ULN Mirror

Localizing the ULN mirror's channel subscriptions prevents the mirror's packages from being updated that would cause package collisions and damage package dependencies.

Ensure that you have subscribed to required channels to serve clients running different OS versions on different platforms, as described in How to Configure the Local ULN Mirror.

This task is required for ULN mirrors that serve heterogeneous clients. In this case, the mirror subscribes to multiple channels, including channels the mirror itself doesn't need. You would need to configure the mirror to prevent its own channel subscriptions from being updated with packages targeted for other clients.

Suppose that the mirror is an Oracle Linux 9 system but is also serving Oracle Linux 8 clients on the x86_64 platform. The following steps would localize the Oracle Linux 9's channel subscriptions:

1. Identify the channels to which the server is subscribed.

   ```
   sudo dnf repolist
   ```

   ```
   ...
   ol8_addons                      Oracle Linux 8 Addons (x86_64)
   ol8_appstream                   Oracle Linux 8 Application Stream
   (x86_64)
   ol8_baseos_latest               Oracle Linux 8 BaseOS Latest (x86_64)
   ...
   ```

   In addition to the system's own Oracle Linux 9 channels, the output would include Oracle Linux 8 channels intended for clients.

2. Edit `/etc/dnf/plugins/spacewalk.conf` to disable repository updates inapplicable to the server.

Use the following format:

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

For the current example, you would specify the following on the file:

```
[ol8_addons]
enabled=0

[ol8_appstream]
enabled=0

[ol8_baseos_latest]
enabled=0
```

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

3. Configure the mirror to be a client of itself.

   See How to Configure Client Access to the Local Mirror.

How to Configure Client Access to the Local Mirror

Clients require access to the local repository mirror to receive updates and errata fixes.

A local mirror must be configured where the clients connect. See previous sections in Using Software Distribution Mirrors.

Perform this task on all the clients in the local network. Use this same procedure to configure the local ULN mirror as a client of itself.

> **Note:**
>
> On Oracle Linux 8 and later clients, use the `dnf` command. On earlier clients, use the `yum` command.

1. Import the GPG key.

   ```
   sudo gpg --import /etc/pki/rpm-gpg/RPM-GPG-KEY
   ```

   The location of the GPG key might differ depending on the Oracle Linux release that's installed on the system. You can also download and import the GPG keys directly from the Oracle Linux yum server. See https://yum.oracle.com/faq.html#a10 for more information.

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

   Choose from one of 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.

     ```
     cd /etc/yum.repos.d
     sudo dnf config-manager|yum-config-manager --disable \*
     ```

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

     ```
     /etc/yum.repos.d> sudo for i in *.repo; do mv $i $i.disabled; done
     ```

3. Create a local `*.repo` file, such as `/etc/yum.repos.d/local-yum.repo`, and populate it with repository entries from the local mirror.

   > **Tip:**
   >
   > To distinguish the local repositories from the public yum repositories or ULN channels, prefix the names of their entries with a string such as `local_`.

   The following example shows entries for an Oracle Linux 8 client:

   ```
   [local_ol8el8_baseos_latest]
   name=Oracle Linux 8 BaseOS Latest ($basearch)
   baseurl=http://local_mirror/repo-location/
   gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
   gpgcheck=1
   enabled=1

   [local_ol8el8_appstream]
   name=Oracle Linux 8 Application Stream ($basearch)
   baseurl=http://local_mirror/repo-location/
   gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
   gpgcheck=1
   enabled=1

   [local_ol8el8_addons]
   name=Oracle Linux 8 Addons ($basearch)
   baseurl=http://local_mirror/repo-location/
   gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
   gpgcheck=1
   enabled=1
   ```

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

   Ensure that the following configurations are correct:

   - All the entries have an `enabled=1` setting.

- The `baseurl` points to the correct mirror location that contains the repositories that each client requires. The locations depend on how you organized the repositories in the mirror's base directory, such as `/var/www/html/yum`.
- The correct GPG key file must exist at the path that's specified for the `gpgkey` parameter. You can download the GPG keys used to sign all the Oracle Linux release packages from the Oracle Linux yum server. See https://yum.oracle.com/faq.html#a10 for more information.

4. Test the configuration.
   a. Clear the yum metadata cache.

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

   b. Verify that the relevant repositories are listed for the client.

      ```
      sudo dnf|yum repolist
      ```

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

5. After confirming that the correct repositories are configured on the client, obtain updates from the local server.

   ```
   sudo dnf|yum update
   ```

7
DNF Command References

The following tables show examples of some of the common tasks that you can perform by using the `dnf` command to manage packages and package groups.

Table 7-1 DNF Commands

Command	Description
`dnf repolist`	Lists all the enabled repositories.
`dnf list`	Lists all the packages that are available in all enabled repositories and all packages that are installed on the system.
`dnf list installed`	Lists all the packages that are installed on the system.
`dnf list available`	Lists all 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`
`dnf check-update`	Checks whether updates exist for packages that are already installed on the 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.

Table 7-1 (Cont.) DNF Commands

Command	Description
`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 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.

Table 7-2 DNF Group Commands

Command	Description
`dnf group list`	Lists Environment Groups, that contain many subgroups; and base groups of packages that are available for installation. To include hidden groups in the list and all the groups' IDs, add the `--hidden -v` options.
`dnf group info` *groupname*	Displays detailed information about a group. If the group is a parent group, this command lists all subgroups that it contains, alternately the command lists all packages that are in the group. To include the groups' IDs, use the `-v` option.
`dnf group install` *groupname*	Installs all the packages in a group.
`dnf group update` *groupname*	Updates all the packages in a group.
`dnf group remove` *groupname*	Removes all the packages in a group.

Table 7-3 DNF Module Commands

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 isn't marked as default or isn't enabled, that package isn't 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 don't 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. As an alternative, you can use `dnf module switch-to`. See the entry in the current table.
`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 don't need to include *stream* and *colon* in the command syntax. Be aware that some modules don't define default streams. As an alternative, you can use `dnf module switch-to`. See the entry in the current table.

Table 7-3 (Cont.) DNF Module Commands

Command Syntax	Description of Action	Additional Information

`dnf install @module-name:stream`
Alternatively, you can use:
`dnf module install module-name:stream`
	Installs a module by using a specific stream and default profiles.	As an alternative, you can use `dnf module switch-to`. See the entry in the current table.

`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.	As an alternative, you can use `dnf module switch-to`. See the entry in the current table.

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

Table 7-3 (Cont.) DNF Module Commands

Command Syntax	Description of Action	Additional Information
`dnf module switch-to @module-name:stream/profile`	Switch to a module stream. This command also changes the versions of installed packages to those versions provided by the new stream and removes packages from the old stream that are no longer available. The command also updates installed profiles if these are available in the new stream.	This command is more encompassing than `dnf module enable` command because it both enables modules and runs `distrosync` on all modular packages in the enabled modules. The command is also more encompassing than `dnf module install` because it both installs profiles and runs `distrosync` on all modular packages in the installed module. However, you must specify a profile with this command because it doesn't use default profiles.
`dnf module remove @module-name`	Removes the specified module.	Removing an installed module removes all the packages and their dependencies that are installed by the profiles of the enabled module stream. The modules to be removed must have some profiles already installed.
`dnf module remove --all @module-name:stream`	Removes all packages from the specified stream.	Note that the command can remove critical packages from the system.
`dnf module remove @module-name:stream/profile`	Removes packages from the installed profile.	
`dnf module remove @module-name:stream`	Removes packages from all installed profiles within the specified stream.	
`dnf module reset @module-name`	Resets a module to the initial state so it is no longer enabled or disabled. Consequently, all installed profiles are removed.	Note that the command does not remove packages from the specified module.
`dnf module disable @module-name`	Disables a module and all its streams. All related module streams become unavailable. Consequently, all installed profiles are removed.	Note that the command does not remove packages from the specified module.

8
Comparing Yum Version 3 With DNF

The following table compares Yum v3 features, commands, and options with the DNF tool that's introduced in Oracle Linux 8.

Yum v3 Feature, Command, or Option	DNF Feature, Command, or Option	Notable Differences
`--skip-broken` option	`--skip-broken` option Is an alias for the `--setopt=strict=0` option	**When used for installations:** Skips all packages (or those with broken dependencies that are passed to DNF) without raising an error or causing the operation to fail. You can use either option with DNF. You can also set this behavior as the default in the `dnf.conf` file. **When used for upgrades:** The semantics that were used to trigger the `yum` command with the`--skip-broken` option are set for `dnf update` as the default. Note that you don't need to use the `--skip-broken` option with the `dnf upgrade` command. Instead, use the `--best` option to use only the latest version of packages in transactions.
`yum update` command	`dnf update` command	Command syntax change only. No differences with the behavior for `dnf update` and`yum update`.
`yum upgrade` command	`dnf upgrade` command	Aside from the syntactical difference, the behavior of `dnf upgrade` is the same as `yum upgrade`. Note that in Yum v3, `yum upgrade` is the same as `yum --obsoletes update`.
`clean_requirements_on_remove` option	`clean_requirements_on_remove` option	This option is enabled by default in DNF, which might cause confusion when comparing the `remove` operation results between the two Yum versions, as DNF removes more packages.

Yum v3 Feature, Command, or Option	DNF Feature, Command, or Option	Notable Differences
`resolvdep` command	Not available Use the `dnf provides` command to find which package provides a specific file.	The Yum v3 command is maintained for legacy purposes *only*.
`deplist` command	Not available Use the `dnfrepoquery--deplist` command to find dependencies for a package.	The `yumdeplist` alias is provided for Yum v3 compatibility with the `dnf repoquery --deplist` command.
Excludes (and repository excludes)	Excludes (and repository excludes)	Yum v3 respects excludes during installations and upgrades; whereas, DNF respects all operations, including erasing, and listing.
`includepkgs` option	`include` option	In DNF, the directive name for repository (and main) configuration has been renamed for better alignment with its DNF counterpart, `exclude` .
`skip_if_available` option	`skip_if_available` option	This option is enabled by default in DNF. Without this setting, and without explicitly setting `skip_if_unavailable=True` in the relevant repository `.ini` file, Yum immediately stops and reports a repository error.
`overwrite_groups` option	Not available	This configuration option has been removed in DNF. Instead, when DNF identifies several groups with the same group ID, it merges the contents of the groups.
`mirrorlist_expire` option	Not available	DNF uses `metadata_expire` for the expiring metadata, and the `mirrorlist` file.
"metalink" mention in the `mirrorlist` repository option.	Not available	A fix has been applied in DNF to render the following information in the `yum.conf(5)` inapplicable: If the `mirrorlist` URL contains the word *metalink*, then the value of `mirrorlist` is copied to `metalink` (if `metalink` isn't set).
`alwaysprompt` option	Not available	This option has been removed from DNF to simplify configuration.

Yum v3 Feature, Command, or Option	DNF Feature, Command, or Option	Notable Differences
`group_package_types` option	Not available	This option has been removed from DNF to simplify configuration.
`upgrade_requirements_on_install`	Behaves as though disabled.	Because DNF tolerates the use of other package managers, it's possible that not all changes that are made to RPMDB are stored in the history of transactions. Thus, DNF doesn't fail in this situation, which means the `force` option is no longer required.
`yum swap` command	`dnfshell` command This command performs a remove and install transaction. `dnf --allowerasing` command	Using the `dnf --allowerasing` command is the same as using `yum swap A B`, where you want to replace `A` (providing `P`) with `B` (also providing `P`), which conflicts with `A`, without removing `C` (which requires `P`).
Dependency processing details displayed during the depsolving phase.	Not available	In DNF, the `depsolver` considers all dependencies for update candidates, which would result in a lengthy output. Note that the Yum v3 output can also be confusing and lengthy, especially for large transactions.
`yum provides` command	`dnf provides` command	The behavior of the `dnf provides` command is aligned to how it's documented; whereas, during the execution of the `yum provides` command, Yum applies certain, undocumented behavior. For example, if you run the `yum provides sandbox` command, Yum applies extra heuristics to interpret the `sandbox` part of the command, then it sequentially prepends entries from the `PATH` environment variable to the command to match a file that's provided by a package. DNF doesn't emulate this undocumented behavior.
`--enableplugins` option	Not available	This option isn't documented for DNF, as all plugins are enabled by default.

Yum v3 Feature, Command, or Option	DNF Feature, Command, or Option	Notable Differences
`throttle` and `bandwidth` options	`throttle` and `bandwidth` options	In DNF, for simultaneous downloads, the total downloading speed is now throttled. This support wasn't available in the Yum v3 tool, as downloaders ran in different processes.
`installonlypkgs` option	`installonlypkgs`	DNF appends the list values from the `installonlypkgs` configuration option to DNF defaults. Yum v3 overwrites the defaults by option values.
`deltarpm_percentage` option	Not available	The Boolean `deltarpm` option controls whether delta RPM files are used. Yum DNF doesn't support the use of the `deltarpm_percentage` option. Instead, the tool chooses the best value of the DRPM/RPM ratio to decide whether using `deltarpm` is appropriate in a particular situation.
`.srpm` files and non-existent package handling	`.srpm` files and non-existent package handling	DNF stops early with an error if a command requesting an installing operation on a local `.srpm` file is run. Yum v3 issues a warning and continues by installing the `tour` package. Note that Yum DNF emits the same error for package specifications that don't match any available package.
Promoting a package to install to a package that obsoletes it.	Promoting a package to install to a package that obsoletes it.	DNF doesn't automatically replace a request to install a package (`A`) by installing another package (`B`) if package B would obsolete package(`A`. The Yum v3 behavior is to perform the action if the `obsoletes` configuration option is enabled. However, note that this behavior isn't documented and can be harmful.

Yum v3 Feature, Command, or Option	DNF Feature, Command, or Option	Notable Differences
`--installroot` option	`--installroot` option	DNF provides more predictable behavior for this option and handles the path differently than the `--config` option, where this path is always related to the host system. Yum v3 combines this path with the `installroot` option. The `reposdir` option is also handled slightly differently in Yum DNF. For example, if one `reposdirs` path exists inside `installroot`, then repositories are taken only from `installroot`. Whereas, Yum v3 tests each path from `reposdir`.
Prompts displayed after a transaction table	Prompts displayed after a transaction table	The prompts that are displayed after a transaction table are different in DNF than they're for Yum v3. DNF doesn't provide download functionality after displaying the transaction table. You're only prompted to continue with the transaction or not. To download packages, use the `download` command.
`list` command	`list` command	The DNF behavior for this command is to list all packages from all repositories, which means there can be duplicate package names with different repository names listed. This change was made to enable users to choose a preferred repository.

There isn't a direct replacement for the `yum-updateonboot` command in DNF. However, you can obtain a similar result by running the `dnfautomatic` command.

The following table compares Yum V3 plugins with DNF plugins.

Yum Version 3 Plugin	DNF Plugin	Package
`yum check`	`dnf repoquery --unsatisfied`	`dnf`
`yum-langpacks`		`dnf-langpacks`
`yum-plugin-auto-update-debug-info`	Option in `debuginfo-install.conf`	`dnf-plugins-core`

Yum Version 3 Plugin	DNF Plugin	Package
yum-plugin-copr	dnf copr	dnf-plugins-core
yum-plugin-fastestmirror	fastestmirror option in dnf.conf	dnf
yum-plugin-fs-snapshot		dnf-plugins-extras-snapper
yum-plugin-local		dnf-plugins-core
yum-plugin-merge-conf		dnf-plugins-extras-rpmconf
yum-plugin-priorities	priority option in dnf.conf	dnf
yum-plugin-remove-with-leaves	dnfautoremove	dnf
yum-plugin-show-leaves		dnf-plugins-core
yum-plugin-versionlock		dnf-plugins-core
yum-rhn-plugin		dnf-plugin-spacewalk

The following table compares Yum v3 utilities with DNF plugins.

Yum Version 3 Utility	DNF Plugin	DNF Package
debuginfo-install	dnf debuginfo-install	dnf-plugins-core
find-repos-of-install	dnf list installed	dnf
needs-restarting	dnf tracer	dnf-plugins-extras-tracer
package-cleanup	dnf list, dnf repoquery	dnf, dnf-plugins-core
repoclosure	dnf repoclosure	dnf-plugins-extras-repoclosure
repodiff	dnf repodiff	dnf-plugins-core
repo-graph	dnf repograph	dnf-plugins-extras-repograph
repomanage	dnf repomanage	dnf-plugins-extras-repomanage
repoquery	dnf repoquery	dnf
reposync	dnf reposync	dnf-plugins-core
repotrack	dnf download –resolve –alldeps	dnf-plugins-core
yum-builddep	dnf builddep	dnf-plugins-core
yum-config-manager	dnf config-manager	dnf-plugins-core

Yum Version 3 Utility	DNF Plugin	DNF Package
`yum-debug-dump`	`dnf debug-dump`	`dnf-plugins-extras-debug`
`yum-debug-restore`	`dnf debug-restore`	`dnf-plugins-extras-debug`
`yumdownloader`	`dnf download`	`dnf-plugins-core`

The following table lists the Yum v3 `package-cleanup` command and its DNF replacement.

Yum Version 3 Command	DNF Command
`package-cleanup--dupes`	`dnfrepoquery--duplicates`
`package-cleanup--leaves`	`dnfrepoquery--unneeded`
`package-cleanup--orphans`	`dnfrepoquery--extras`
`package-cleanup--oldkernels`	`dnfrepoquery--installonly`
`package-cleanup--problems`	`dnfrepoquery--unsatisfied`
`package-cleanup--cleandupes`	`dnfremove--duplicates`
`package-cleanup--oldkernels`	`dnfremove--oldinstallonly`

A

Application Stream Module Life Cycle

While the core operating system packages in the BaseOS repository for Oracle Linux 8 and Oracle Linux 9 retain a standard Oracle Linux support life cycle, the separate AppStream packages have their own major version releases and might have 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.

See Oracle Linux: Product Life Cycle Information for more information on the Application Stream Life Cycle durations for Oracle Linux releases.

B
Managing ULN Users

You must have a least one valid Customer Support Identifier (CSI) to access ULN. The CSI is an identifier that's issued to you when you buy 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 can manage and administer CSIs against user accounts and systems from within ULN.

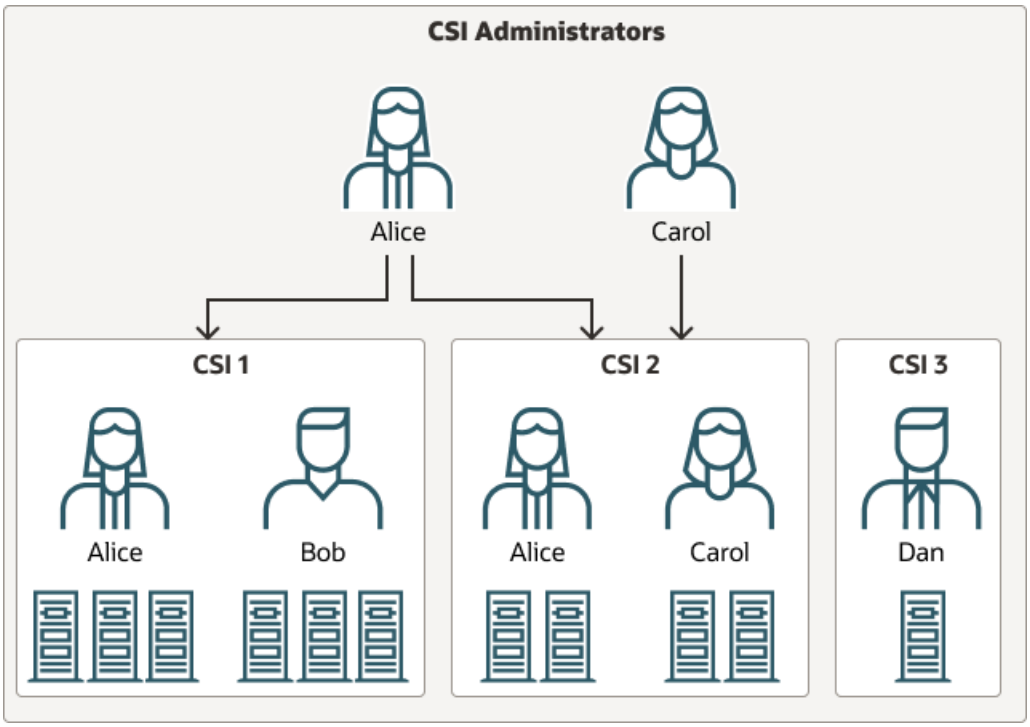
The CSI administration feature of ULN provides a unified view of all an organization's CSIs and the systems that are registered with those CSIs. To manage the registered systems, you must become an administrator for one or more of your organization's CSIs. To view and change the details of any system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

If you're 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 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 Listing Active CSIs and Transfer 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. SeeListing Expired CSIs and Transferring Their Registered Servers.
- Remove yourself or someone else as administrator of a CSI. See Removing a CSI Administrator.

Figure B-1 shows a representative example of an organization with three CSIs, only two of which have CSI administrators.

Figure B-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.

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

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

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

Dan isn't registered as an administrator for CSI 3. Dan can view only the details of the system that's registered to Dan's account in CSI 3.

Neither Bob nor Dan can perform CSI administration tasks. For example, they can't move systems between CSIs nor can they reassign systems to other users. However, as CSI 3 doesn't currently have an administrator, Dan can choose to become its

administrator. As CSI 1 already has Alice as its administrator, Bob can't become an administrator unless Alice grants that privilege.

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

Registering to Use ULN

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. See How to Register a System With ULN for more information.

You can register with ULN without registering a system. To use ULN without first registering a system, you can register as a ULN user if you have a valid customer support identifier (CSI) for Oracle Linux support or Oracle VM support. This can be useful to track errata and security information, regardless of whether any of your systems are registered with ULN.

To buy Oracle Linux or Oracle VM support, go to the online Oracle Linux Store or contact a sales representative.

To register as a ULN user:

1. In a browser, go to https://linux.oracle.com/register.
2. If you don't 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 username 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're prompted to click **Confirm** to become the CSI administrator. If you click **Cancel**, you can't access the CSI administration feature. See Becoming a CSI Administrator.
>
> If your username already exists on the system, you're prompted to proceed to ULN by clicking the link **Unbreakable Linux Network**. If you enter a different CSI from your existing CSIs, your username is associated with the new CSI in addition to your existing CSIs.

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're prompted to click **Confirm** to become the CSI administrator. If you click **Cancel**, you can't 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're registered, you're 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're already an administrator of a CSI, you can add yourself as administrator of another CSI if you have registered either a server or your ULN username with the other CSI.

  To assign yourself as administrator of another 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. The page lists any other administrators, if they exist, 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's next to the username of the user that you want to add as an administrator. Their username is added to the **Administrator** box.
  5. If you administer more than one CSI, select the CSI that the user can administer from the **CSI** drop down list.
  6. Click **Assign Administrator**.

> **Note:**
>
> To become the administrator of a CSI, but the person to whom it's registered is no longer with your organization, contact an Oracle support representative to request that you be made the administrator for the CSI.

Listing Active CSIs and Transfer 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:
>
> - Their username must be registered to this CSI.
> - One or more of the servers, for which they're the owner, must be registered to this CSI.
> - They must be an administrator of at least one CSI for which you're 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's next to the username 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's 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.

Listing Expired CSIs and Transferring Their Registered Servers

To list details of the expired CSIs for which you're 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:
>
> - Their username must be registered to this CSI.
> - One or more of the servers, for which they're the owner, must be registered to this CSI.
> - They must be an administrator of at least one CSI for which you're 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's next to the username 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's 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.

Removing a CSI Administrator

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's next to the username 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**.

C

About 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 Authentication Methods.

`channel`
Contains methods for listing software channels on ULN. See Channel Methods.

`channel.software`
Contains methods for querying packages available within different channels on ULN. See Channel Software Methods.

`errata`
Contains methods for interacting with errata on ULN. See Errata Methods.

`packages`
Contains methods for querying package information for specified packages on ULN. See Packages Methods.

`system`
Contains methods for managing systems registered with ULN. See System Methods.

Authentication Methods

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

- `auth.login`
- `auth.logout`

`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`

This key is active until you call the `auth.logout` method.

`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.

Channel Methods

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

- `channel.listSoftwareChannels`

`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_64_latest`

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.

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.

- channel.software.getDetails
- channel.software.listAllPackages
- channel.software.listErrata
- channel.software.listLatestPackages

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: `abc4ef3d6e6b2bc3246e56ee4756ed5c245b60b0`

string file_name

The file name for the filelists metadata at the channel location. For example: `repodata/filelists.xml.gz`

string url

The URL where the filelists metadata can be accessed.

To access the URL, include the `X-ULN-Api-User-Key` header with the value of the session key that was returned when you authenticated. For example:

```
curl -H "X-ULN-Api-User-Key:
JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc" \
https://uln.oracle.com/XMLRPC/GET-REQ/ol7_x86_64_latest/
repodata/filelists.xml.gz
```

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: `90acbe6860bbcd4e40ee71cec9d2397dceccbca6`

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.

To access the URL, include the `X-ULN-Api-User-Key` header with the value of the session key that was returned when you authenticated. For example:

```
curl -H "X-ULN-Api-User-Key:
JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc" \
https://uln.oracle.com/XMLRPC/GET-REQ/ol7_x86_64_latest/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:
`20f6b193cd9376d650cf96c8c01995cf7f02163a`

string file_name

The file name for the other metadata at the channel location. For example:
`repodata/other.xml.gz`

string url

The URL where the other metadata can be accessed.

To access the URL, include the `X-ULN-Api-User-Key` header with the value of the session key that was returned when you authenticated. For example:

```
curl -H "X-ULN-Api-User-Key:
JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc" \
https://uln.oracle.com/XMLRPC/GET-REQ/ol7_x86_64_latest/repodata/
other.xml.gz
```

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:
`3992e1e77d476d09eb1dcb16fd106263aaa84bb4`

string file_name

The file name for the primary metadata at the channel location. For example:
`repodata/primary.xml.gz`

string url

The URL where the primary metadata can be accessed.

To access the URL, include the `X-ULN-Api-User-Key` header with the value of the session key that was returned when you authenticated. For example:

```
curl -H "X-ULN-Api-User-Key:
JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc" \
https://uln.oracle.com/XMLRPC/GET-REQ/ol7_x86_64_latest/
repodata/primary.xml.gz
```

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.

To access the URL, include the `X-ULN-Api-User-Key` header with the value of the session key that was returned when you authenticated. For example:

```
curl -H "X-ULN-Api-User-Key:
JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc" \
https://uln.oracle.com/XMLRPC/GET-REQ/ol7_x86_64_latest/
repodata/repomd.xml
```

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: `6d11ecbceb58515be79a2adff9ff911f8a839069`

string file_name

The file name for the updateinfo metadata at the channel location. For example: `repodata/updateinfo.xml.gz`

string url

The URL where the updateinfo metadata can be accessed.

To access the URL, include the `X-ULN-Api-User-Key` header with the value of the session key that was returned when you authenticated. For example:

```
curl -H "X-ULN-Api-User-Key:
JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc" \
https://uln.oracle.com/XMLRPC/GET-REQ/ol7_x86_64_latest/repodata/
updateinfo.xml.gz
```

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 sequentially. For example: `1`

  string package_id

  The package ID within the ULN infrastructure. For example: `11776733`

  string package_last_modified

  The date and timestamp for when a package was last modified. For example: `2018-09-27 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_5.6`

string package_version
The package version number. For example: `3.13.1`

`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-2018-4255`

  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: `glibc bug fix update`

  string errata_update_date
  The errata update date. For example: `2018-10-17 00:00:00`

`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 infrastructure. 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.el7_5.6`

string package_version

The package version number. For example: `3.13.1`

Errata Methods

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

- `errata.applicableToChannels`
- `errata.getDetails`
- `errata.listCves`
- `errata.listPackages`

`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_latest`

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.

`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: `[0:1.2.1-7.3]\n- Add missing connection hostname check against X.509 certificate name\n- Resolves: CVE-2012-5784`

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`

`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`

`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

URL value.

To access a URL, include the `X-ULN-Api-User-Key` header with the value of the session key that was returned when you authenticated. For example:

```
curl -H "X-ULN-Api-User-Key:
JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc" \
https://uln.oracle.com/XMLRPC/GET-REQ/ol7_x86_64_latest/
java-1.8.0-openjdk-demo-1.8.0.191.b12-0.el7_5.i686.rpm
```

array providing_channels

An array listing channels providing this package.

string label

A string with the channel label as a value. For example:
`ol7_x86_64_latest`

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:
`2018-10-17 16:39:10`

string package_build_host

For example: `x86-ol7-builder-02.us.oracle.com`

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-1.8.0.191.b12-0.el7_5.i686.rpm`

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 advertising and GPL+ and GPLv2 and GPLv2 with exceptions 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: `1508de7bafe0d6fe0814d216cbbb354b`

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: `1.8.0.191.b12`

struct package_checksums

A structure, listing package checksum values by type:

string md5

The md5 hash for the package checksum value. For example: `1508de7bafe0d6fe0814d216cbbb354b`

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:

- `packages.getDetails`
- `packages.listProvidingErrata`

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

  URL value.

  To access a URL, include the `X-ULN-Api-User-Key` header with the value of the session key that was returned when you authenticated. For example:

  ```
  curl -H "X-ULN-Api-User-Key:
  JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc" \
  https://uln.oracle.com/XMLRPC/GET-REQ/ol7_x86_64_latest/
  java-1.8.0-openjdk-demo-1.8.0.191.b12-0.el7_5.i686.rpm
  ```

array providing_channels

An array listing channels providing this package.

string label

A string with the channel label as a value. For example: `ol7_x86_64_latest`

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: `2018-10-17 16:39:10`

string package_build_host

The host where the package was built. For example: `x86-ol7-builder-02.us.oracle.com`

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-1.8.0.191.b12-0.el7_5.i686.rpm`

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 advertising and GPL+ and GPLv2 and GPLv2 with exceptions 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: `1508de7bafe0d6fe0814d216cbbb354b`

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: `1.8.0.191.b12`

struct package_checksums
A structure, listing package checksum values by type:

string md5
The md5 hash for the package checksum value. For example: `1508de7bafe0d6fe0814d216cbbb354b`

`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-2018-2942`

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: `2018-10-17 00:00:00`

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:

- `system.deleteSystems`

`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.