Oracle Linux

Unbreakable Linux Network User's Guide

G38926-01
August 2025

Oracle Linux Unbreakable Linux Network User's Guide,

G38926-01

Contents

5 Using ULN Software Distribution Mirrors

6 Managing ULN Users

A About the ULN API

Preface

Oracle Linux: Unbreakable Linux Network User's Guide describes how to manage software on Oracle Linux systems using the Unbreakable Linux Network (ULN). This guide explains how to register systems with ULN, subscribe to software channels, and keep systems current with updates, security patches, and packages that aren't available through the public Oracle Linux yum server. The information in this guide helps administrators maintain secure, reliable, and up-to-date Oracle Linux environments using ULN.

Documentation License

Conventions

The following text conventions are used in this document:

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

Documentation Accessibility

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

Access to Oracle Support for Accessibility

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

Diversity and Inclusion

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

1

Introduction to Unbreakable Linux Network (ULN)

This chapter introduces the Unbreakable Linux Network (ULN), Oracle's service for delivering Oracle Linux software updates, security patches, and extra packages through logically organized channels. It also explains how ULN fits alongside other Oracle Linux software distribution options, and highlights key differences between yum repositories and ULN channels to help administrators get started with software management in enterprise environments.

> **Important**
>
> The General Oracle Linux documentation no longer describes how to subscribe to ULN channels. As a ULN user, you need to understand the Relationship Between Yum Repositories and ULN Channels.

Oracle Linux Package Distribution Options

Oracle Linux systems can obtain software and updates through several mechanisms. The main options are:

- **Oracle Linux Yum Server**: The primary, publicly available yum server at https://yum.oracle.com/ , which provides free access to many Oracle Linux packages.
- **Unbreakable Linux Network (ULN)**: A protected, customer-only network that distributes Oracle Linux software channels, some of which aren't available through public yum servers. ULN also lets you access extra software such as Oracle Ksplice, tools for other Oracle products, and features to manage patches and updates.
- **Oracle Cloud Infrastructure (OCI) Yum Servers**: In OCI environments, all ULN channels are replicated to regional yum servers, providing direct access to all Oracle Linux software packages from OCI compute instances in a familiar yum repository format.

> **Important**
>
> For OCI users and those managing their on-premises or third-party cloud systems with OS Management Hub, all ULN channels are mirrored as yum repositories. Registration with ULN isn't required for these systems.
>
> For more information on OS Management Hub, see Oracle OS Management Hub.

The rest of this guide focuses solely on ULN. For more information about the other Oracle Linux package distribution options, see Oracle Linux: Managing Software on Oracle Linux.

About ULN

The Unbreakable Linux Network (ULN) is Oracle's comprehensive service for managing software updates, patches, and channel-based repositories for Oracle Linux systems. To use ULN, you must be an Oracle Linux Support customer.

ULN provides software in organized groups called *channels*. Each channel contains packages for a particular Oracle Linux version, system architecture, or purpose. Administrators select which channels a system uses to control what software and updates that system can receive.

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 information about `yum`, `dnf`, Ksplice, and support policies. The ULN Alert Notification Tool periodically checks with ULN and alerts you when updates are available.

To access the ULN's web interface, visit https://linux.oracle.com/ .

Relationship Between Yum Repositories and ULN Channels

Both yum repositories and ULN channels are directories of packages, along with their relevant metadata, which are accessed by clients using the `dnf` or `yum` command line tools. However, the packages follow a different naming scheme:

- **Yum repositories** are named based on the Oracle Linux release and software stream. For example, `ol10_appstream` or `ol10_baseos_latest` provide the minimum required packages for Oracle Linux 10.
- **ULN channels** include the platform architecture in their name, for example the `ol10_x86_64_appstream` and `ol10_aarch64_baseos_latest` channels provide the minimum required packages for Oracle Linux 10 on systems with x86_64 architectures. For aarch64 systems, the `ol10_aarch64_appstream` and `ol10_aarch64_baseos_latest` channels provide the minimum required packages for Oracle Linux 10.

> **Tip**
>
> These naming conventions are important when translating instructions: if documentation instructs you to enable a yum repository, then instead subscribe the system to the corresponding ULN channel for that system's architecture using the ULN web interface.

Accessing ULN

To access ULN, you must be an Oracle Linux Support customer with a valid Customer Support 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 get started with ULN, create a ULN profile.

To use `dnf` with ULN, you must first generate an authentication token that can be used to access ULN and then 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.

2

ULN Channels

Channels correspond to the architecture of a system. The Unbreakable Linux Network has hundreds of unique channels. These enable access to the packages for all releases of Oracle Linux. ULN channels are available for different platform architectures, such as x86_64, and the 64-bit Arm architectures. ULN channels also exist for MySQL, Oracle Ksplice, OCFS2, RDMA, and productivity applications. Other channels might also become available, such as unsupported channels for developer preview, or for specific developer content.

ULN Channel Types

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 Managing a System's Channel Subscriptions. See also Main ULN Channels.

Main ULN Channels

Oracle Linux 10

The following table lists the primary ULN channels for Oracle Linux 10. Extra channels are available. Check the ULN web interface for a complete list. The corresponding yum repositories are available on https://yum.oracle.com/.

Channel	Description
ol10_*arch*_baseos_latest	Core channel. Provides all the latest versions of the base OS 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.
ol10_*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.
ol10_*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.
ol10_*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 ol10_*arch*_appstream channel. Support for the codeready_builder packages is limited to package installation help only.
ol10_*arch*_developer	Provides packages intended for developers to create test and development environments for Oracle Linux 10 and related technologies. Support for the developer packages is limited to package installation help only.

Channel	Description
ol10_u*n*_*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 help only. Unlike in previous releases of Oracle Linux, the EPEL channels are synced with the update level of the release, for example, ol10_u0_x86_64_developer_EPEL.

Oracle Linux 9

The following table lists the primary ULN channels for Oracle Linux 9. Extra channels are available. Check the ULN web interface for a complete list. The corresponding yum repositories are available on https://yum.oracle.com/.

Channel	Description
ol9_*arch*_baseos_latest	Core channel. Provides all the latest versions of the base OS 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.

Channel	Description
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 help only.
ol9_*arch*_developer	Provides packages intended for developers to create test and development environments for Oracle Linux 9 and related technologies. Support for the developer packages is limited to package installation help only.
ol9_*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 help only.

Oracle Linux 8

The following table lists the primary ULN channels for Oracle Linux 8. Extra channels are available. Check the ULN web interface for a complete list. The corresponding yum repositories are available on https://yum.oracle.com/.

Channel	Description
ol8_*arch*_baseos_latest	Core channel Provides all the latest versions of the base OS 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.

Channel	Description
ol8_*arch*_un_baseos_base	Provides the base versions of the base OS 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 aren't provided in this channel. To keep the system updated and secure, subscribe to the appropriate _baseos_patch channel or subscribe to the appropriate _baseos_latest channel. If the system is subscribed to the _baseos_latest channel, you don't need to subscribe to this channel.
ol8_*arch*_un_baseos_patch	Provides the patched versions of the base OS 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. To keep a system updated and secure, subscribe to the appropriate _baseos_latest channel. If the system is subscribed to the _baseos_latest channel, you don't 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.
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 help 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 help only.

Channel	Description
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 help only.

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 About ULN.

Registering a System With ULN

Registering a system with ULN provides 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.
- You must have a ULN profile configured for your SSO account. See Creating a ULN Profile for more information.
- You must have a valid authentication token for ULN. See Generating an Authentication Token for more information.
- 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.

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. Use the ULN Authentication Token as the password, when 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. Select the **ULN Registration** shortcut icon.

c. Provide the credentials as prompted. Use the ULN Authentication Token as the password, when prompted.

The Set Up Software Updates window is displayed.

2. Select **Next**.
3. Provide the ULN username, authentication token, and customer support identifier (CSI).
4. Enter a ULN name for the system.
5. Select 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.

Managing 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. Subscribing to a ULN channel is equivalent to enabling a yum repository.

Ensure that the system is registered with ULN. See Registering a System With ULN.

1. Sign in to https://linux.oracle.com with valid SSO credentials.
2. (Optional) View the available channels to which you can subscribe the system.
   a. Select 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. Select 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, select **Manage Subscriptions**.
   c. On the System Summary page, select channels from the list of available or subscribed channels and select 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, select **Save Subscriptions**.

Changing 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.

1. Sign in to https://linux.oracle.com with valid SSO credentials.
2. On the Systems tab, select the link named for the system in the list of registered machines.
3. On the System Details page, select **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, select **Apply Changes**.

Removing 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.

1. Sign in to https://linux.oracle.com with valid SSO credentials.
2. On the Systems tab, select the link named for the system in the list of registered machines.
3. On the System Details page, select **Delete**.
4. To confirm the deletion, select **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.

Configuring a System to Use a Proxy for ULN

To configure a system to use a proxy server when accessing ULN, update the proxy configuration settings in /etc/sysconfig/rhn/up2date.

- Edit /etc/sysconfig/rhn/up2date and add the enableProxy and httpProxy configuration variables as required.

  ```
  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 system password, and especially not the root user password.

4

Updating Software on Oracle Linux Using ULN

After you have enabled the appropriate ULN channels, software updates are achieved using standard `dnf` commands on the system. For more information on using the `dnf` utility in Oracle Linux, see: Oracle Linux: Managing Software on Oracle Linux.

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.

> **Important**
>
> After performing a system update where many packages are updated, we recommend that you reboot the system. System functionality might become unstable if core packages are updated and the system isn't restarted to load the most recent updates.

Tracking Security Updates and Errata Releases

Oracle releases important changes to the Oracle Linux 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-*`.

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.

To be notified when new errata packages are released, you can subscribe to the relevant 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.

> **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 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. Sign in to https://linux.oracle.com with the appropriate SSO credentials.
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, select the title of the **Type**, **Severity**, **Advisory**, **Systems Affected**, or **Release Date** column. Select the title again to reverse the order of sorting.

     > **Note**
     >
     > The **Systems Affected** column shows how many systems might be affected by an advisory.

   - To display or hide advisories of different types, select or clear the **Bug**, **Enhancement**, and **Security** check boxes and select **Go**.
   - To display only advisories for a certain release of Oracle Linux or Oracle VM, select that release from the **Release** list and select **Go**.
   - To search within the table, enter a string in the **Search** field and select **Go**.
4. To see more detail about an advisory and to download the RPMs:
   a. Select 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're browsing.

Using 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. Sign in to https://linux.oracle.com with valid SSO credentials.
2. On the Systems tab, select 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. Select **Download All Available Errata for this System**.

   Or, 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. Select 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.

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.

5

Using ULN Software Distribution Mirrors

Distribution mirrors are alternative sources of software packages to the channels on Unbreakable Linux Network (ULN). These are selected channels that you locally replicate from ULN. 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 ULN channels for systems that don't have access to a public network.
- Improve software download times and reduce bandwidth overhead for larger infrastructure.
- Set up network-based installation infrastructure.
- Work with 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 ULN channels. The channels can be made available to client systems in the internal network through various methods such as using a local web server or a file transfer server.

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 the wider infrastructure.

> **Note**
>
> For customers with OCI access, we recommend using OS Management Hub as an alternative and more feature rich approach to providing mirrored software resources available on ULN to local systems. See Oracle OS Management Hub for more information.

Prerequisities for a ULN Local Mirror

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

- Must have Internet access to connect to ULN.
- Must have enough memory to create the yum metadata.
- Must be configured to provide access to the mirrored channels by system clients.
- Must have 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 channels that you decide to mirror. Other factors are the number of clients to be serviced, including their platforms, OS, and other specific packages that each client might require 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 channels are also updated on a regular schedule and further affect storage requirements on the local server.

For guidance in estimating the disk size requirements, use the appropriate tools to calculate the size of a specific channel. Because channels are dynamic and grow over time, always plan to allocate substantially greater disk space than the current size suggests. Optionally, you can create a dedicated file system and mount this to the directory that hosts the mirrored channels.

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 or 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 public Oracle Linux yum server. Where possible, set up mirrors of Oracle Linux yum server repositories instead.

Configuring the Local ULN Mirror

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

The ULN mirror must meet the requirements described in Prerequisities for a ULN Local Mirror. Also, you must have:

- Set up the system as a distribution mirror.
- Generated a ULN Authentication Token
- 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, sign in at https://linux.oracle.com using valid SSO credentials.
b. On the Systems tab, select the link named for the system chosen to be a ULN mirror.
c. On the System Details page, select **Edit**.
d. On the Edit System Properties page, select the **Yum Server** checkbox.
e. Select **Apply Changes**.

- **Using the `uln-channel` command**

a. At a terminal prompt, type:

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

b. If prompted, specify the appropriate ULN username and authentication token.

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 chosen ULN mirror, select **Manage Subscriptions**.
b. On the System Summary page, select channels from the list of available or subscribed channels and select 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, select **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 username and authentication token.
c. (Optional) To verify that the subscriptions completed successfully, type:

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

3. Create a protected and unprotected version of `/etc/dnf/plugins/spacewalk.conf`. For example:

```
sudo cp /etc/dnf/plugins/spacewalk.conf /etc/dnf/plugins/spacewalk.conf.protected
sudo cp /etc/dnf/plugins/spacewalk.conf /etc/dnf/plugins/spacewalk.conf.unprotected
```

Edit `/etc/dnf/plugins/spacewalk.conf.protected` to disable channels that you aren't using for the ULN mirror itself. See Localizing Subscriptions for the ULN Mirror Server. This procedure disables the channels that don't apply to the system.

4. Switch to the unprotected version of /etc/dnf/plugins/spacewalk.conf that enables all channels required for the mirror. For example:

```
cp /etc/dnf/plugins/spacewalk.conf.unprotected /etc/dnf/plugins/spacewalk.conf
```

5. Mirror the ULN Channels to the location of the base directory for the mirror. For example:

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

You can use the --repoid and --exclude options with the reposync command to control exactly which repositories you're mirroring and to help reduce disk space requirements by excluding source packages. For example:

```
sudo dnf reposync --delete --download-metadata -p /var/www/html/yum \
  --repoid ol10_x86_64_ksplice \
  --exclude *.src,*.nosrc
sudo dnf reposync --delete --download-metadata -p /var/www/html/yum \
  --repoid ol10_x86_64_userspace_ksplice \
   --exclude *.src,*.nosrc
sudo dnf reposync --delete --download-metadata -p /var/www/html/yum \
  --repoid ol9_x86_64_ksplice \
  --exclude *.src,*.nosrc
sudo dnf reposync --delete --download-metadata -p /var/www/html/yum \
  --repoid ol9_x86_64_userspace_ksplice \
   --exclude *.src,*.nosrc
```

6. Switch to the protected version of /etc/dnf/plugins/spacewalk.conf that enables only those channels used by the system hosting the mirror. For example:

```
sudo cp /etc/dnf/plugins/spacewalk.conf.protected /etc/dnf/plugins/spacewalk.conf
```

7. Consider creating a cron script or systemd service and timer to update the mirror at intervals. For example, create a file at /etc/cron.daily/uln-mirror-update that automatically switches to the unprotected version of /etc/dnf/plugins/spacewalk.conf, updates the mirror based on the channels enabled, then switches back to the protected version of /etc/dnf/plugins/spacewalk.conf. If the system is configured to use itself as a mirror, the local yum repository configuration must be disabled while the mirror is updated:

```
#!/bin/bash
####### Regularly update yum repos ######

# Change DNF configuration to allow all repositories
cp /etc/dnf/plugins/spacewalk.conf.unprotected /etc/dnf/plugins/spacewalk.conf

# Check whether the system is configured as a client of itself
if [ -f /etc/yum.repos.d/local-yum.repo ]; then
   mv /etc/yum.repos.d/local-yum.repo /etc/yum.repos.d/local-yum.repo.disabled;
fi

# Run the reposync. You can change this command to specify the
# repoid and exclusions that you want for a more customized mirror

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

```
# Change DNF configuration to use protected repositories
cp /etc/dnf/plugins/spacewalk.conf.protected /etc/dnf/plugins/spacewalk.conf

# Enable the yum configuration again
if [ -f /etc/yum.repos.d/local-yum.repo.disabled ]; then
  mv /etc/yum.repos.d/local-yum.repo.disabled /etc/yum.repos.d/local-yum.repo;
fi
```

Ensure that the file is executable. For example:

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

Localizing Subscriptions for the ULN Mirror Server

Localizing the ULN mirror's channel subscriptions for the server hosting the mirror prevents the mirror's packages incompatible with the host system 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 Configuring the Local ULN Mirror.

If the mirror serves clients running different OS versions or hardware architectures, subscribe to all required channels, including channels the host server itself doesn't need. Configure the host server to prevent its own channel subscriptions from being updated with packages targeted for other clients by disabling those channels locally.

Suppose that the server hosting the mirror is an Oracle Linux 10 system but the mirror is also serving Oracle Linux 9 clients on the x86_64 platform. The following steps would localize the Oracle Linux 10's channel subscriptions such that only those channels applicable to Oracle Linux 10 are enabled when not updating the ULN mirror:

1. Identify the channels the server is subscribed to.

   ```
   sudo dnf repolist
   ```

   ```
   ol9_x86_64_ksplice      Ksplice for Oracle Linux 9 (x86_64)
   ol9_x86_64_userspace_ksplice      Ksplice-aware user space packages for Oracle Linux 9 (x86_64)
   ol9_x86_64_ksplice              Ksplice for Oracle Linux 10 (x86_64)
   ol10_x86_64_userspace_ksplice     Ksplice-aware user space packages for Oracle Linux 10 (x86_64)
   ol10_x86_64_UEKR8               Oracle Linux 10 UEK Release 8 (x86_64)
   ol10_x86_64_appstream           Oracle Linux 10 Application Stream Packages (x86_64)
   ol10_x86_64_baseos_latest       Oracle Linux 10 BaseOS  Latest(x86_64)
   ```

   In addition to the system's own Oracle Linux 10 channels, the output would include Oracle Linux 9 channels intended for clients.

2. Edit /etc/dnf/plugins/spacewalk.conf.protected to disable channel updates inapplicable to the server and enable those that are applicable.

   Use the following format:

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

For the current example, you would disable all Oracle Linux 9 channels:

```
[ol9_x86_64_ksplice]
enabled = 0

[ol9_x86_64_userspace_ksplice]
enabled = 0

[ol10_x86_64_ksplice]
enabled = 1

[ol10_x86_64_userspace_ksplice]
enabled = 1

[ol10_x86_64_UEKR8]
enabled = 1

[ol10_x86_64_appstream]
enabled = 1

[ol10_x86_64_baseos_latest]
enabled = 1
```

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

3. Edit /etc/dnf/plugins/spacewalk.conf.unprotected to enable all channels intended for the mirror.

   Use the following format:

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

   For the current example, you would enable both the Oracle Linux 9 and Oracle Linux 10 channels:

   ```
   [ol9_x86_64_ksplice]
   enabled = 1

   [ol9_x86_64_userspace_ksplice]
   enabled = 1

   [ol10_x86_64_ksplice]
   enabled = 1

   [ol10_x86_64_userspace_ksplice]
   enabled = 1

   [ol10_x86_64_UEKR8]
   enabled = 1
   ```

```
[ol10_x86_64_appstream]
enabled = 1

[ol10_x86_64_baseos_latest]
enabled = 1
```

6
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.

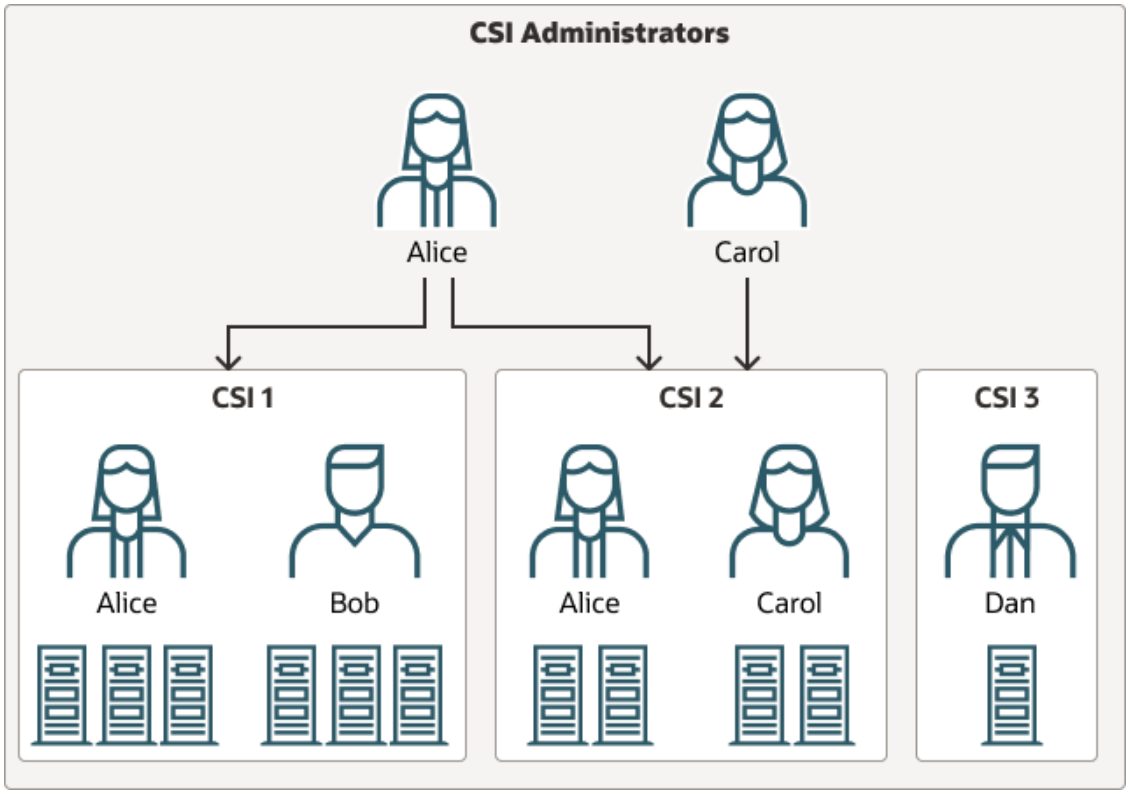
The CSI administration feature of ULN provides a unified view of all an organization's CSIs and the systems that are registered with those CSIs. To manage the registered systems, you must become an administrator for one or more of your organization's CSIs. To view and change the details of any system 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 Transferring Their Registered Servers.
- List expired CSIs, list the servers that are currently registered with an expired CSI, and transfer those servers to another user or to another CSI. SeeListing Expired CSIs and Transferring Their Registered Servers.
- Remove yourself or someone else as administrator of a CSI. See Removing a CSI Administrator.

Figure 6-1 shows a representative example of an organization with three CSIs, only two of which have CSI administrators.

Figure 6-1 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.

Creating a ULN Profile

Create a ULN Profile so that you can start interacting with ULN and register systems.

To use ULN you must first create a ULN profile by registering as a ULN user. You can register as a ULN user if you have a valid customer support identifier (CSI) for Oracle Linux support or Oracle VM support.

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 already have an Oracle Account, select **Sign On**.

   Sign in using valid Oracle SSO credentials.

   If you don't have an Oracle Account, select **Create New Account** and follow the onscreen instructions to create one.
3. On the Create New ULN User page, enter the CSI, and select **Create New User**.

> **Note**
>
> If no administrator is assigned to manage the CSI, you're prompted to select **Confirm** to become the CSI administrator. If you select **Cancel**, you can't access the CSI administration feature. See Becoming a CSI Administrator.
>
> If the username already exists in ULN, you're prompted to proceed to ULN by selecting 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.

Generating an Authentication Token

Generate a ULN service authentication token to use for ULN registration and other authenticated interactions, such as connecting to the API.

Authentication tokens, or secret keys, are used instead of password authentication to help protect ULN and to limit exposure of Oracle SSO credentials. You can use the authentication token instead of your password when accessing ULN using the uln-channel, uln_register, or ulnreg_ks commands, or when using the ULN API.

To generate or regenerate an authentication token:

1. In a web browser, sign in to ULN at https://linux.oracle.com/ using your SSO credentials.
2. Select the **Auth Token** option in the header navigation.
3. On the Auth Token page, select **Generate Secret Key** and note down the secret key.

The secret key is only displayed once, during the initial generation.

You can select **Copy Secret Key** to copy the key to the clipboard.

If you lose or forget the secret key, generate a new one by revisiting the Auth Token page and select **Delete Secret Key**, then select **Generate Secret Key** again.

Becoming a CSI Administrator

You can become an administrator of a CSI in several different ways.

> **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.

- When you register with ULN, if no administrator is assigned to manage the CSI, you're prompted to select **Confirm** to become the CSI administrator. If you select **Cancel**, you can't access the CSI administration feature.
- When signed in to ULN, if you access the System tab and no administrator is assigned to manage one of the CSIs for which you're registered, you're prompted to select whether to become the CSI administrator.
  1. Select 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 select **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 a ULN username with the other CSI.
  1. Sign in to ULN and select the CSI Administration tab.
  2. On the Managed CSIs page, select **Add CSI**.
  3. On the Assign Administrator page, enter the CSI, and select **Add**.
  4. The page lists any other administrators, if they exist, and prompts you to select **Confirm** to confirm the request.
- An administrator for a CSI can add you as an administrator for the same CSI.
  1. Sign in to ULN as administrator of the CSI, and select the CSI Administration tab.
  2. On the Managed CSIs page, select **List Administrators**.
  3. On the CSI Administrators page, select **Assign Administrator**.
  4. On the Assign Administrator page in the Select New Administrator list, select the **+** icon that's next to the username of the user that you want to add as an administrator.
  5. If you administer more than one CSI, select the CSI that the user can administer from the **CSI** dropdown list.

6. Select **Assign Administrator**.

Listing Active CSIs and Transferring Their Registered Servers

Steps to list active CSIs and transfer their registered servers to another user or CSI.

> **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.

1. Sign 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. Select 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.
   a. Select the **Transfer System** check boxes for the systems that you want to transfer to another user.
   b. Select **Transfer Selected Systems to Another Owner**.
   c. On the **Transfer Registered System(s) - Owner** page in the **Transfer To** column, select 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, select **Apply Changes** to confirm the transfer to the new owner.
5. To transfer systems to another CSI, follow these steps:
   a. Select the **Transfer System** check boxes for the systems that you want to transfer to another CSI.
   b. Select **Transfer Selected Systems to Another CSI**.
   c. On the **Transfer Registered System(s) - CSI** page in the **Transfer To** column, select 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, select **Apply Changes** to confirm the transfer to the new CSI.

The selected systems are successfully transferred to either another user or another CSI, based on the selections made during the process.

Listing Expired CSIs and Transferring Their Registered Servers

Steps to list expired CSIs and transfer their registered servers to another user or CSI.

> **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.

1. Sign 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. Select 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.

   To transfer systems to another user:

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

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

   c. On the **Transfer Registered System(s) - Owner** page in the **Transfer To** column, select 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, select **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. Select **Transfer Selected Systems to Another CSI**.

   c. On the **Transfer Registered System(s) - CSI** page in the **Transfer To** column, select 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, select **Apply Changes** to confirm the transfer to the new CSI.

The selected systems are transferred to either another user or another CSI, depending on the selections made during the process.

Removing a CSI Administrator

Steps to remove a CSI administrator.

1. Sign in to ULN and select the **CSI Administration** tab.
2. On the **Managed CSIs** page, select **List Administrators**.
3. On the **CSI Administrators** page in the **Delete?** column, select 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, select **OK**.

The selected user no longer has administrative privileges for the specified CSI.

A

About the ULN API

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

The Unbreakable Linux Network (ULN) API is an XML-RPC interface that lets you programmatically access, manage, and automate ULN functions such as channel subscriptions, package queries, and errata retrieval for Oracle Linux systems.

This API is based on XML-RPC, which lets applications perform remote operations by encoding the procedure calls in XML and sending 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 signs in to ULN using a specified username and authentication token.

Input Parameters

Parameter	Description
username	The Oracle Account username to use for the session.
password	The authentication token to use for the session. See Generating an Authentication Token for more information.

Return Parameters

Parameter	Description
sessionKey	The session key for the session. All other methods use the session key during the session. This key is active until you call the auth.logout method.

Example Input Parameters

username: example.user@oracle.com
password: MCWvP+0meWmwJmL7esd

Example Output Parameters

JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc

auth.logout

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

Input Parameters

Parameter	Description
sessionKey	The session key of the session to be terminated. For example: JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc

Return Parameters

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

Example Input Parameters

sessionKey: uyjAN3cB7ySsAUra27CKj3qx9fuGyio5uQ6w4q4ALNX

Example Output Parameters

1

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

Parameter	Description
sessionKey	The session key for the session.

Return Parameters

Field	Description
array	An array of channels with: **struct (channel)** **channel_arch** The channel architecture. **channel_end_of_life** The channel end of life. Currently unused on ULN. **channel_label** The channel label. **channel_name** The channel name. **channel_parent_label** The channel parent label. Currently unused on ULN.

Example Input Parameters

sessionKey: uyjAN3cB7ySsAUra27CKj3qx9fuGyio5uQ6w4q4ALNX

Example Output Parameters

```
[
  {
    "channel_arch": "x86_64",
    "channel_name": "Enterprise Linux 4 Update 5 Patch (x86_64)",
    "channel_end_of_life": " ",
    "channel_label": "el4_u5_x86_64_patch",
    "channel_parent_label": " "
  },
  {
    "channel_arch": "x86_64",
    "channel_name": "Enterprise Linux 4 Update 4 Patches (x86_64)",
    "channel_end_of_life": " ",
    "channel_label": "el4_u4_x86_64_patch",
    "channel_parent_label": " "
  }
]
```

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

Parameter	Description
sessionKey	The session key for the session.
channelLabel	The channel label for the channel that you want to query.

Return Parameters

Field	Description
channel_arch_name	The channel architecture name.
channel_description	The channel description.
channel_summary	The channel summary, usually the same as the channel name.

Field	Description
metadata_urls	A dictionary or associative array of metadata locations and checksum information, including the URLs to download channel metadata. **filelists** **checksum_type** The hashing algorithm used to generate the checksum. **checksum** The checksum for the filelists metadata file. **file_name** The file name for the filelists metadata at the channel location. **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. **group** **checksum_type** The hashing algorithm used to generate the checksum. **checksum** The checksum for the group metadata file. **file_name** The file name for the group metadata at the channel location. **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. **other** **checksum_type** The hashing algorithm used to generate the checksum. **checksum** The checksum for the other metadata file. **file_name** The file name for the other metadata at the channel location. **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. **primary** **checksum_type** The hashing algorithm used to generate the checksum.

Field	Description
	checksum The checksum for the primary metadata file. **file_name** The file name for the primary metadata at the channel location. **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. **repomd** **file_name** The file name for the repomd metadata at the channel location. **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. **updateinfo** **checksum_type** The hashing algorithm used to generate the checksum. **checksum** The checksum for the updateinfo metadata file. **file_name** The file name for the updateinfo metadata at the channel location. **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.

Example Input Parameters

sessionKey: uyjAN3cB7ySsAUra27CKj3qx9fuGyio5uQ6w4q4ALNX
channelLabel: ol10_x86_64_baseos_latest

Example Output Parameters

{
  "channel_description": "Base operating system packages released for Oracle Linux 10 (x86_64) including the latest errata packages",
  "channel_summary": "Oracle Linux 10 BaseOS Latest (x86_64)",
  "metadata_urls": {
    "group": [
      {

```
            "url": "https://linux-update.oracle.com/XMLRPC/GET-REQ/ol10_x86_64_baseos_latest/repodata/
a02a929c88a276cf81b88ff672457550172a5a28-comps.xml",
            "checksum": "a02a929c88a276cf81b88ff672457550172a5a28",
            "file_name": "repodata/a02a929c88a276cf81b88ff672457550172a5a28-comps.xml",
            "checksum_type": "sha1"
          }
        ],
        "filelists": [
          {
            "url": "https://linux-update.oracle.com/XMLRPC/GET-REQ/ol10_x86_64_baseos_latest/repodata/
3cbd811376b2e4aeb061786723ae7d8f3bd4de16-filelists.xml.gz",
            "checksum": "3cbd811376b2e4aeb061786723ae7d8f3bd4de16",
            "file_name": "repodata/3cbd811376b2e4aeb061786723ae7d8f3bd4de16-filelists.xml.gz",
            "checksum_type": "sha1"
          }
        ],
        "updateinfo": [
          {
            "url": "https://linux-update.oracle.com/XMLRPC/GET-REQ/ol10_x86_64_baseos_latest/repodata/
b15bbd51b9077e12e798a906ef353900b8d6b018-updateinfo.xml.gz",
            "checksum": "b15bbd51b9077e12e798a906ef353900b8d6b018",
            "file_name": "repodata/b15bbd51b9077e12e798a906ef353900b8d6b018-updateinfo.xml.gz",
            "checksum_type": "sha1"
          }
        ],
        "group_gz": [
          {
            "url": "https://linux-update.oracle.com/XMLRPC/GET-REQ/ol10_x86_64_baseos_latest/repodata/
13b7d0bf3fa83daa9133dcc9c1759c182a931678-comps.xml.gz",
            "checksum": "13b7d0bf3fa83daa9133dcc9c1759c182a931678",
            "file_name": "repodata/13b7d0bf3fa83daa9133dcc9c1759c182a931678-comps.xml.gz",
            "checksum_type": "sha1"
          }
        ],
        "primary": [
          {
            "url": "https://linux-update.oracle.com/XMLRPC/GET-REQ/ol10_x86_64_baseos_latest/repodata/
5c6108529bf4d5c3bc916aa56d3ebb86a447f43b-primary.xml.gz",
            "checksum": "5c6108529bf4d5c3bc916aa56d3ebb86a447f43b",
            "file_name": "repodata/5c6108529bf4d5c3bc916aa56d3ebb86a447f43b-primary.xml.gz",
            "checksum_type": "sha1"
          }
        ],
        "repomd": [
          {
            "url": "https://linux-update.oracle.com/XMLRPC/GET-REQ/ol10_x86_64_baseos_latest/repodata/
repomd.xml",
            "file_name": "repodata/repomd.xml"
          }
        ],
        "other": [
          {
            "url": "https://linux-update.oracle.com/XMLRPC/GET-REQ/ol10_x86_64_baseos_latest/repodata/
45576c9eefa12f1fe1188044763be5a56f765699-other.xml.gz",
            "checksum": "45576c9eefa12f1fe1188044763be5a56f765699",
            "file_name": "repodata/45576c9eefa12f1fe1188044763be5a56f765699-other.xml.gz",
```

```
            "checksum_type": "sha1"
         }
      ],
      "other_db": [
         {
            "url": "https://linux-update.oracle.com/XMLRPC/GET-REQ/ol10_x86_64_baseos_latest/repodata/
ff29e8fc2fb34c5ccf162b9799a518c00372794f-other.sqlite.bz2",
            "checksum": "ff29e8fc2fb34c5ccf162b9799a518c00372794f",
            "file_name": "repodata/ff29e8fc2fb34c5ccf162b9799a518c00372794f-other.sqlite.bz2",
            "checksum_type": "sha1"
         }
      ],
      "primary_db": [
         {
            "url": "https://linux-update.oracle.com/XMLRPC/GET-REQ/ol10_x86_64_baseos_latest/repodata/
a17b11542d6b426a3a115bf3b144ef2394430a06-primary.sqlite.bz2",
            "checksum": "a17b11542d6b426a3a115bf3b144ef2394430a06",
            "file_name": "repodata/a17b11542d6b426a3a115bf3b144ef2394430a06-primary.sqlite.bz2",
            "checksum_type": "sha1"
         }
      ],
      "filelists_db": [
         {
            "url": "https://linux-update.oracle.com/XMLRPC/GET-REQ/ol10_x86_64_baseos_latest/repodata/
ac1c90f41e8e81e0348a4a83ab069bb19e59a79b-filelists.sqlite.bz2",
            "checksum": "ac1c90f41e8e81e0348a4a83ab069bb19e59a79b",
            "file_name": "repodata/ac1c90f41e8e81e0348a4a83ab069bb19e59a79b-filelists.sqlite.bz2",
            "checksum_type": "sha1"
         }
      ]
   },
   "channel_arch_name": "x86_64"
}
```


channel.software.listAllPackages

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

Input Parameters

Parameter	Description
sessionKey	The session key for the session. For example: JyUVNoT74BFaRJ6fRjDIQ5idPmCaj5UJLb76E2f45Gc
channelLabel	The channel label for the channel that you want to query.

Return Parameters

Field	Description
array	An array of all packages: **struct (package)** **package_arch_label** The package architecture label. For example: noarch **package_epoch** The package epoch value, if specified. The epoch value can help RPM decide package version ordering if the versioning doesn't make sense or doesn't follow sequentially. For example: 1 **package_id** The package ID within the ULN infrastructure. For example: 11776733 **package_last_modified** The date and timestamp for when a package was last modified. For example: 2018-09-27 19:31:13 **package_name** The name of the package. For example: selinux-policy-mls **package_release** The package release information. For example: 192.0.6.el7_5.6 **package_version** The package version number. For example: 3.13.1

Example Input Parameters

sessionKey: uyjAN3cB7ySsAUra27CKj3qx9fuGyio5uQ6w4q4ALNX
channelLabel: ol10_x86_64_baseos_latest

Example Output Parameters


```
[
   {
      "package_arch_label": "x86_64",
      "package_name": "ethtool",
      "package_epoch": "2",
      "package_version": "6.11",
      "package_release": "5.el10_0",
      "package_id": 33833622,
      "package_last_modified": "2025-07-01 13:59:36"
   },
   {
      "package_arch_label": "x86_64",
```

```
    "package_name": "oraclelinux-release",
    "package_epoch": "10",
    "package_version": "10.0",
    "package_release": "1.0.15.el10",
    "package_id": 33833623,
    "package_last_modified": "2025-06-27 22:22:26"
  }
]
```


channel.software.listErrata

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

Input Parameters

Parameter	Description
sessionKey	The session key for the session.
channelLabel	The channel label for the channel that you want to query.

Return Parameters

Field	Description
array	An array of all errata associated with the channel label: **struct (errata)** **errata_advisory_type** The errata advisory type. **errata_advisory** The errata advisory label. **errata_issue_date** The date the errata was issued. **errata_last_modified_date** The date the errata was last modified. **errata_synopsis** A brief synopsis of the errata. **errata_update_date** The errata update date.

Example Input Parameters

sessionKey: uyjAN3cB7ySsAUra27CKj3qx9fuGyio5uQ6w4q4ALNX
channelLabel: ol10_x86_64_baseos_latest

Example Output Parameters


```
[
  {
    "errata_update_date": "2025-07-22 00:00:00",
    "errata_advisory_type": "Bug Fix Advisory",
    "errata_synopsis": "sos bug fix and enhancement update",
    "errata_advisory": "ELBA-2025-11538",
    "errata_last_modified_date": "2025-07-22 00:00:00",
    "errata_issue_date": "2025-07-22 00:00:00"
  },
  {
    "errata_update_date": "2025-07-22 00:00:00",
    "errata_advisory_type": "Security Advisory",
    "errata_synopsis": "kernel security update",
    "errata_advisory": "ELSA-2025-11428",
    "errata_last_modified_date": "2025-07-22 00:00:00",
    "errata_issue_date": "2025-07-22 00:00:00"
  }
]
```


channel.software.listLatestPackages

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

Input Parameters

Parameter	Description
sessionKey	The session key for the session.
channelLabel	The channel label for the channel that you want to query.

Return Parameters

Field	Description
array	An array of latest packages: **struct (package)** **package_arch_label** The package architecture label. **package_epoch** The package epoch value, if specified. The epoch value can help RPM decide package version ordering if the versioning doesn't make sense or doesn't follow sequentially. **package_id** The package ID within the ULN infrastructure. **package_name** The name of the package. **package_release** The package release information. **package_version** The package version number.

Example Input Parameters

sessionKey: uyjAN3cB7ySsAUra27CKj3qx9fuGyio5uQ6w4q4ALNX
channelLabel: ol10_x86_64_baseos_latest

Example Output Parameters


```
[
   {
      "package_arch_label": "x86_64",
      "package_name": "libcomps",
      "package_epoch": " ",
      "package_version": "0.1.21",
      "package_release": "3.el10",
      "package_id": 33658475
   },
   {
      "package_arch_label": "x86_64",
      "package_name": "fuse-common",
      "package_epoch": " ",
      "package_version": "3.16.2",
      "package_release": "5.el10",
      "package_id": 33658505
   }
```

]

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

Parameter	Description
sessionKey	The session key for the session.
advisoryName	The name of the erratum.

Return Parameters

Field	Description
array	An array of channels: **struct (channel)** **channel_id** The identifier for a channel in the ULN infrastructure. **channel_label** The label for the channel. **channel_name** The full name for the channel. **parent_channel_id** The parent channel id. Not currently used on ULN.

Example Input Parameters

sessionKey: uyjAN3cB7ySsAUra27CKj3qx9fuGyio5uQ6w4q4ALNX
advisoryName: ELSA-2025-11428

Example Output Parameters


```
[
   {
      "channel_name": "Oracle Linux 10 Application Stream Packages (aarch64)",
      "channel_label": "ol10_aarch64_appstream",
      "parent_channel_id": " ",
      "channel_id": 12056
   },
   {
      "channel_name": "Oracle Linux 10 Application Stream Packages (x86_64)",
      "channel_label": "ol10_x86_64_appstream",
      "parent_channel_id": " ",
      "channel_id": 12057
   }
]
```


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

Parameter	Description
sessionKey	The session key for the session.
advisoryName	The name of the erratum.

Return Parameters

Field	Description
array	An array of detailed information associated with the erratum: **struct (erratum)** **errata_description** The detailed description of the erratum. **errata_issue_date** The date the erratum was issued. **errata_last_modified_date** The date the erratum was last modified. **errata_notes** Notes associated with the erratum. Usually empty. **errata_references** References of the erratum. Usually empty. **errata_severity** The severity level set for the erratum. **errata_synopsis** A brief synopsis of the erratum. **errata_topic** The topic for the erratum. Usually empty. **errata_type** The type for the erratum. **errata_update_date** The errata update date.

Example Input Parameters

sessionKey: uyjAN3cB7ySsAUra27CKj3qx9fuGyio5uQ6w4q4ALNX
advisoryName: ELSA-2025-11428

Example Output Parameters


```
{
   "errata_update_date": "7/22/25",
   "errata_topic": " ",
   "errata_type": "Security Advisory",
   "errata_severity": "Important",
   "errata_notes": "",
   "errata_synopsis": "kernel security update",
   "errata_references": " ",
```

"errata_last_modified_date": "2025-07-22 00:00:00",
"errata_issue_date": "7/22/25",
"errata_description": "[6.12.0-55.22.1.0.1_0.OL10]
- nvme-pci: remove two deallocate zeroes quirks [Orabug: 37756650]
- Add new Oracle Linux Driver Signing (key 1) certificate [Orabug: 37985782]
- Disable UKI signing [Orabug: 36571828]
- Update Oracle Linux certificates (Kevin Lyons)
- Disable signing for aarch64 (Ilya Okomin)
- Oracle Linux RHCK Module Signing Key was added to the kernel trusted keys list (olkmod_signing_key.pem) [Orabug: 29539237]
- Update x509.genkey [Orabug: 24817676]
- Conflict with shim-ia32 and shim-x64 <= 15.3-1.0.5]
- Remove upstream reference during boot (Kevin Lyons) [Orabug: 34729535]
- Add Oracle Linux IMA certificates
- Update module name for cryptographic module [Orabug: 37400433]"
}

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

Parameter	Description
sessionKey	The session key for the session.
advisoryName	The name of the erratum.

Return Parameters

Field	Description
array	An array of CVE IDs. If no matching CVE IDs are found, the array is empty: **cve_name** The CVE ID associated with the erratum ID.

Example Input Parameters

sessionKey: uyjAN3cB7ySsAUra27CKj3qx9fuGyio5uQ6w4q4ALNX
advisoryName: ELSA-2025-11428

Example Output Parameters

[
"CVE-2025-38089",
"CVE-2025-21905"

]

errata.listPackages

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

Input Parameters

Parameter	Description
sessionKey	The session key for the session.
advisoryName	The name of the erratum.

Return Parameters

Field	Description
array	An array of packages: **struct (package)** **download_urls** An array of URLs where the package can be downloaded from: **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. **providing_channels** An array listing channels providing this package: **label** A string with the channel label as a value. **package_arch_label** The package architecture label. **package_build_date** The date and timestamp for when the package was built. **package_build_host** The host where the package was built. **package_cookie** The package cookie value. Usually empty. **package_description** The full description of the package. **package_epoch** The package epoch value, if specified. The epoch value can help RPM decide package version ordering if the versioning doesn't make sense or doesn't follow sequentially. **package_file** The package filename. **package_id** The package ULN identifier. **package_last_modified_date** The date and timestamp for when the package was last modified. **package_license** The license or licenses that a package is released under. **package_md5sum** The package md5sum value.

Field	Description
	package_name The package name. **package_payload_size** The package payload size in bytes. **package_release** The package release value. **package_size** The package size in bytes. **package_summary** A summary of the contents of the package. **package_vendor** The package vendor name. **package_version** The package version. **package_checksums** A structure, listing package checksum values by type: **md5** The md5 hash for the package checksum value.

Example Input Parameters

sessionKey: uyjAN3cB7ySsAUra27CKj3qx9fuGyio5uQ6w4q4ALNX
advisoryName: ELSA-2025-11428

Example Output Parameters


```
[
  {
    "package_last_modified_date": "2025-07-22 12:46:21",
    "package_arch_label": "aarch64",
    "package_cookie": " ",
    "package_name": "python3-perf",
    "package_summary": "Python bindings for apps which will manipulate perf events",
    "package_epoch": " ",
    "package_checksums": [
      {
        "sha256": "f11de229c7b58dbe4fb67b09725fade3a1cf82aa3bb333f3d546d0b7a7eeb6af"
      }
    ],
    "package_payload_size": 3350148,
    "package_version": "6.12.0",
    "package_license": "((GPL-2.0-only WITH Linux-syscall-note) OR BSD-2-Clause) AND ((GPL-2.0-only
```

WITH Linux-syscall-note) OR BSD-3-Clause) AND ((GPL-2.0-only WITH Linux-syscall-note) OR CDDL-1.0) AND ((GPL-2.0-only WITH Linux-syscall-note) OR Linux-OpenIB) AND ((GPL-2.0-only WITH Linux-syscall-note) OR MIT) AND ((GPL-2.0-or-later WITH Linux-syscall-note) OR BSD-3-Clause) AND ((GPL-2.0-or-later WITH Linux-syscall-note) OR MIT) AND 0BSD AND BSD-2-Clause AND (BSD-2-Clause OR Apache-2.0) AND BSD-3-Clause AND BSD-3-Clause-Clear AND CC0-1.0 AND GFDL-1.1-no-invariants-or-later AND GPL-1.0-or-later AND (GPL-1.0-or-later OR BSD-3-Clause) AND (GPL-1.0-or-later WITH Linux-syscall-note) AND GPL-2.0-only AND (GPL-2.0-only OR Apache-2.0) AND (GPL-2.0-only OR BSD-2-Clause) AND (GPL-2.0-only OR BSD-3-Clause) AND (GPL-2.0-only OR CDDL-1.0) AND (GPL-2.0-only OR GFDL-1.1-no-invariants-or-later) AND (GPL-2.0-only OR GFDL-1.2-no-invariants-only) AND (GPL-2.0-only WITH Linux-syscall-note) AND GPL-2.0-or-later AND (GPL-2.0-or-later OR BSD-2-Clause) AND (GPL-2.0-or-later OR BSD-3-Clause) AND (GPL-2.0-or-later OR CC-BY-4.0) AND (GPL-2.0-or-later WITH GCC-exception-2.0) AND (GPL-2.0-or-later WITH Linux-syscall-note) AND ISC AND LGPL-2.0-or-later AND (LGPL-2.0-or-later OR BSD-2-Clause) AND (LGPL-2.0-or-later WITH Linux-syscall-note) AND LGPL-2.1-only AND (LGPL-2.1-only OR BSD-2-Clause) AND (LGPL-2.1-only WITH Linux-syscall-note) AND LGPL-2.1-or-later AND (LGPL-2.1-or-later WITH Linux-syscall-note) AND (Linux-OpenIB OR GPL-2.0-only) AND (Linux-OpenIB OR GPL-2.0-only OR BSD-2-Clause) AND Linux-man-pages-copyleft AND MIT AND (MIT OR Apache-2.0) AND (MIT OR GPL-2.0-only) AND (MIT OR GPL-2.0-or-later) AND (MIT OR LGPL-2.1-only) AND (MPL-1.1 OR GPL-2.0-only) AND (X11 OR GPL-2.0-only) AND (X11 OR GPL-2.0-or-later) AND Zlib AND (copyleft-next-0.3.1 OR GPL-2.0-or-later)",
      "package_vendor": "Oracle America",
      "package_release": "55.22.1.0.1.el10_0",
      "package_size": 3348903,
      "package_sha256": "f11de229c7b58dbe4fb67b09725fade3a1cf82aa3bb333f3d546d0b7a7eeb6af",
      "package_id": 33986233,
      "providing_channels": [
        "ol10_aarch64_appstream"
      ],
      "package_build_host": "build-ol10-aarch64.oracle.com",
      "package_description": "The python3-perf package contains a module that permits applications
written in the Python programming language to use the interface
to manipulate perf events.",
      "package_build_date": "2025-07-22 12:46:21",
      "download_urls": [
        "https://linux-update.oracle.com/XMLRPC/GET-REQ/ol10_aarch64_appstream/python3-perf-6.12.0-55.22.1.0.1.el10_0.aarch64.rpm"
      ],
      "package_file": "python3-perf-6.12.0-55.22.1.0.1.el10_0.aarch64.rpm"
    },
    {
      "package_last_modified_date": "2025-07-22 12:46:21",
      "package_arch_label": "aarch64",
      "package_cookie": " ",
      "package_name": "rtla",
      "package_summary": "Real-Time Linux Analysis tools",
      "package_epoch": " ",
      "package_checksums": [
        {
          "sha256": "2a0dd56416265790b06929b7b8396f3b95b7cd239c730baf1b8e6499920e26b8"
        }
      ],
      "package_payload_size": 160948,
      "package_version": "6.12.0",
      "package_license": "((GPL-2.0-only WITH Linux-syscall-note) OR BSD-2-Clause) AND ((GPL-2.0-only WITH Linux-syscall-note) OR BSD-3-Clause) AND ((GPL-2.0-only WITH Linux-syscall-note) OR CDDL-1.0) AND ((GPL-2.0-only WITH Linux-syscall-note) OR Linux-OpenIB) AND ((GPL-2.0-only WITH Linux-syscall-note) OR MIT) AND ((GPL-2.0-or-later WITH Linux-syscall-note) OR BSD-3-Clause) AND ((GPL-2.0-or-later WITH Linux-syscall-note) OR MIT) AND 0BSD AND BSD-2-Clause AND (BSD-2-Clause OR Apache-2.0) AND

BSD-3-Clause AND BSD-3-Clause-Clear AND CC0-1.0 AND GFDL-1.1-no-invariants-or-later AND GPL-1.0-or-later AND (GPL-1.0-or-later OR BSD-3-Clause) AND (GPL-1.0-or-later WITH Linux-syscall-note) AND GPL-2.0-only AND (GPL-2.0-only OR Apache-2.0) AND (GPL-2.0-only OR BSD-2-Clause) AND (GPL-2.0-only OR BSD-3-Clause) AND (GPL-2.0-only OR CDDL-1.0) AND (GPL-2.0-only OR GFDL-1.1-no-invariants-or-later) AND (GPL-2.0-only OR GFDL-1.2-no-invariants-only) AND (GPL-2.0-only WITH Linux-syscall-note) AND GPL-2.0-or-later AND (GPL-2.0-or-later OR BSD-2-Clause) AND (GPL-2.0-or-later OR BSD-3-Clause) AND (GPL-2.0-or-later OR CC-BY-4.0) AND (GPL-2.0-or-later WITH GCC-exception-2.0) AND (GPL-2.0-or-later WITH Linux-syscall-note) AND ISC AND LGPL-2.0-or-later AND (LGPL-2.0-or-later OR BSD-2-Clause) AND (LGPL-2.0-or-later WITH Linux-syscall-note) AND LGPL-2.1-only AND (LGPL-2.1-only OR BSD-2-Clause) AND (LGPL-2.1-only WITH Linux-syscall-note) AND LGPL-2.1-or-later AND (LGPL-2.1-or-later WITH Linux-syscall-note) AND (Linux-OpenIB OR GPL-2.0-only) AND (Linux-OpenIB OR GPL-2.0-only OR BSD-2-Clause) AND Linux-man-pages-copyleft AND MIT AND (MIT OR Apache-2.0) AND (MIT OR GPL-2.0-only) AND (MIT OR GPL-2.0-or-later) AND (MIT OR LGPL-2.1-only) AND (MPL-1.1 OR GPL-2.0-only) AND (X11 OR GPL-2.0-only) AND (X11 OR GPL-2.0-or-later) AND Zlib AND (copyleft-next-0.3.1 OR GPL-2.0-or-later)",
    "package_vendor": "Oracle America",
    "package_release": "55.22.1.0.1.el10_0",
    "package_size": 159073,
    "package_sha256": "2a0dd56416265790b06929b7b8396f3b95b7cd239c730baf1b8e6499920e26b8",
    "package_id": 33986228,
    "providing_channels": [],
    "package_build_host": "build-ol10-aarch64.oracle.com",
    "package_description": "The rtla meta-tool includes a set of commands that aims to analyze
the real-time properties of Linux. Instead of testing Linux as a black box,
rtla leverages kernel tracing capabilities to provide precise information
about the properties and root causes of unexpected results.",
    "package_build_date": "2025-07-22 12:46:21",
    "download_urls": [],
    "package_file": "rtla-6.12.0-55.22.1.0.1.el10_0.aarch64.rpm"
  }
]

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

Parameter	Description
sessionKey	The session key for the session.
pid	The package identifier that should be queried, specified as an integer.

Return Parameters

Field	Description
array	An array of channels with: **struct (package)** **download_urls** An array of URLs where the package can be downloaded from. **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. **providing_channels** An array listing channels providing this package. **label** A string with the channel label as a value. **package_arch_label** The package architecture label. **package_build_date** The date and timestamp for when the package was built. **package_build_host** The host where the package was built. **package_cookie** The package cookie value. Usually empty. **package_description** The full description of the package. **package_epoch** The package epoch value, if specified. The epoch value can help RPM decide package version ordering if the versioning doesn't make sense or doesn't follow sequentially. **package_file** The package filename. **package_id** The ULN package identifier. **package_last_modified_date** The date and timestamp for when the package was last modified. **package_license** The license or licenses that a package is released under. **package_md5sum** The package md5sum value.

Field	Description
	package_name The package name. **package_payload_size** The package payload size in bytes. **package_release** The package release value. **package_size** The package size in bytes. **package_summary** A summary of the contents of the package. **package_vendor** The package vendor name. **package_version** The package version. **package_checksums** A structure, listing package checksum values by type: **md5** The md5 hash for the package checksum value.

Example Input Parameters

sessionKey: uyjAN3cB7ySsAUra27CKj3qx9fuGyio5uQ6w4q4ALNX
pid: 33658475

Example Output Parameters


```
{
  "package_last_modified_date": "2025-01-26 17:00:08",
  "package_arch_label": "x86_64",
  "package_cookie": " ",
  "package_name": "libcomps",
  "package_summary": " ",
  "package_epoch": " ",
  "package_checksums": [
    {
      "sha256": "d62a4818769711fb742b4309b581de8f1572c601c338a7bc3a4802393e11e89b"
    }
  ],
  "package_payload_size": 211496,
  "package_version": "0.1.21",
  "package_license": "GPL-2.0-or-later",
  "package_vendor": " ",
```

```
   "package_release": "3.el10",
   "package_size": 210223,
   "package_sha256": "d62a4818769711fb742b4309b581de8f1572c601c338a7bc3a4802393e11e89b",
   "package_id": 33658475,
   "providing_channels": [
      "ol10_x86_64_baseos_latest"
   ],
   "package_build_host": "build-ol10-x86_64.oracle.com",
   "package_description": " ",
   "package_build_date": "2025-01-26 17:00:08",
   "download_urls": [
      "https://linux-update.oracle.com/XMLRPC/GET-REQ/ol10_x86_64_baseos_latest/
libcomps-0.1.21-3.el10.x86_64.rpm"
   ],
   "package_file": "libcomps-0.1.21-3.el10.x86_64.rpm"
}
```


packages.listProvidingErrata

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

Input Parameters

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

Return Parameters

Field	Description
array	An array of all errata associated with the package: **struct (errata)** **errata_advisory_type** The errata advisory type. For example: Security Advisory **errata_advisory** The errata advisory label. For example: ELSA-2018-2942 **errata_issue_date** The date the errata was issued. For example: 2018-10-17 00:00:00 **errata_last_modified_date** The date the errata was last modified. For example: 2018-10-17 00:00:00 **errata_synopsis** A brief synopsis of the errata. For example: java-1.8.0-openjdk security update **errata_update_date** The errata update date. For example: 2018-10-17 00:00:00

Example Input Parameters

sessionKey: uyjAN3cB7ySsAUra27CKj3qx9fuGyio5uQ6w4q4ALNX
pid: 33658475

Example Output Parameters


```
[
  {
    "errata_update_date": "2025-06-09 00:00:00",
    "errata_advisory_type": "Bug Fix Advisory",
    "errata_synopsis": "libcomps bug fix and enhancement update",
    "errata_advisory": "ELBA-2025-6382",
    "errata_last_modified_date": "2025-06-09 00:00:00",
    "errata_issue_date": "2025-06-09 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

Parameter	Description
sessionKey	The session key for the session.
serverId	The system identifier that should be removed. This needs to be the id value within ULN.

Return Parameters

Field	Description
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.

Example Input Parameters

sessionKey: eMNanSFF7Hp5qEONUr88L8zfeVQTyzmzKx22Rs30eD0
serverId: 330213

Example Output Parameters

0