Introducing Junos OS Evolved

Release
19.1R1

Modified: 2019-06-14

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Introducing Junos OS Evolved
19.1R1

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at https://support.juniper.net/support/eula/. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

Chapter 1 **Overview of Junos OS Evolved 9**

Junos OS Evolved Overview 9
- Benefits 9
- Native Linux Base 9
- Central Database for State 10
- Modular Design 10
- Distributed Infrastructure 11

How the CLI in Junos OS Evolved Differs from Junos OS 11
- New CLI Statements and Commands (Junos OS Evolved) 11
- Modified CLI Statements and Commands (Junos OS Evolved) 13
- Changed CLI Command Output (Junos OS Evolved) 16
- Removed CLI Statements and Commands (Junos OS Evolved) 18

Where to Find Information on Common Procedures 19

Chapter 2 **Booting Junos OS Evolved from a USB 21**

Booting Junos OS Evolved by Using a Bootable USB Drive 21
- Create a Bootable USB Drive Using a Windows Device 21
- Create a Bootable USB Drive Using a MAC OS X 22
- Create a Bootable USB Drive Using a Switch or Router Running Junos OS Evolved 23
- Boot Junos OS Evolved from a Bootable USB Drive 24
- Recover Junos OS Evolved Using USB Scratch Install 25

Chapter 3 **Running Third-Party Applications on Junos OS Evolved 27**

Running Third-Party Applications in Containers 27
- Deploying an LXC Container 27
- Deploying a Docker Container 28

Running Linux Tools on Junos OS Evolved 28
- Example of a Preloaded Linux Command 29
- Interface Name Translation 32
- Other Caveats for the Intercept Feature 34

List of Figures

Chapter 1 Overview of Junos OS Evolved 9
Figure 1: Publish-Subscribe Model 10

List of Tables

Chapter 1 Overview of Junos OS Evolved 9

Table 1: New CLI Statements and Commands (Junos OS Evolved) 11
Table 2: Modified CLI Statements and Commands (Junos OS Evolved) 13
Table 3: Changed Command Output (Junos OS Evolved) 16
Table 4: Removed CLI Statements and Commands (Junos OS Evolved) 18

Chapter 3 Running Third-Party Applications on Junos OS Evolved 27

Table 5: Translation Rule for Interface Names 33
Table 6: Examples of Translated Logical Interface Names 33

CHAPTER 1

Overview of Junos OS Evolved

- Junos OS Evolved Overview on page 9
- How the CLI in Junos OS Evolved Differs from Junos OS on page 11
- Where to Find Information on Common Procedures on page 19

Junos OS Evolved Overview

Junos OS Evolved is next generation Junos OS. It is used just like Junos OS—the same CLI user interface, the same applications and features, the same management and automation tools—but its infrastructure is entirely modernized, giving customers the resiliency, portability, faster innovation, and simplified upgrades they need.

Benefits

The development of Junos OS Evolved provides several benefits to Juniper Networks customers:

- Nearly all of the CLI and user interfaces are identical to those provided in Junos OS. This means there is virtually no learning curve in using Junos OS Evolved.
- It runs natively on Linux, providing direct access to all the Linux utilities and operations.
- All statistics and states are modeled and all states can be uniformly accessed. There is a central database which are used by not only Junos native applications but also external applications (using APIs).
- It has a fully distributed general-purpose software infrastructure that leverages all the compute resources on the network element, for example, CPUs in the Routing Engines, CPUs in the line cards, and potentially other x86 CPUs attached.

Native Linux Base

Whereas Junos OS runs over an instance of the FreeBSD operating system on a specific hardware element (for example, the CPU on the Routing Engine), Junos OS Evolved runs over a native Linux system. Having Linux as a base leverages a much wider, dynamic, and active development community. The Linux system also contains multiple third-party applications and tools developed for Linux that Junos OS Evolved can integrate with minimal effort.

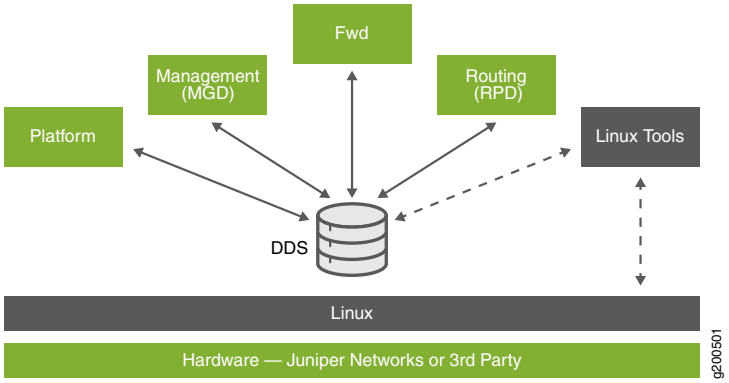
The Junos OS Evolved infrastructure is a horizontal software layer that decouples the application processes from the hardware on which they run. Effectively, this creates a general-purpose software infrastructure spanning all the different compute resources on the system (Routing Engine CPUs, line card CPUs, and possibly others). Application processes (protocols, services, and so on) run on top of this infrastructure and communicate with each other by publishing and consuming (that is, subscribing to) state.

Central Database for State

State is the retained information or status about physical or logical entities that is preserved, shared across the system, and supplied during restarts. State includes both operational and configuration state, including committed configuration, and interface state, routes, and hardware state. In Junos OS Evolved, all state is held in a central database called the Distributed Data Store (DDS).

The DDS does not interpret state. Its only job is to hold state received from subscribers and propagate state to consumers. It implements the publish-subscribe messaging pattern for communicating state between applications that are originators of a state to applications that are consumers of that state (see Figure 1 on page 10). Each application publishes state to and subscribes to state from the DDS directly, making applications independent of each other.

Figure 1: Publish-Subscribe Model



Decoupling applications in this manner isolates the failure of one application from others. The failing application can restart using the last known state of the system held in the state database.

Modular Design

Junos OS Evolved is composed of components with well-defined interfaces. This modular design means you replace at a component level, resulting in flexible packaging and faster upgrades. You can upgrade the system using an application-level restart, which, depending on the changes, in many instances will not require a system reboot. The system determines which modules have changed, and only changed modules are upgraded and restarted. Restarted applications reload the state that is preserved in the DDS.

Distributed Infrastructure

In Junos OS Evolved, a node is an entity composed of a physical compute resource such as a Routing Engine CPU or Line Card CPU or other x86 compute resource. Each node runs base Linux OS, state distribution pub-sub infrastructure, and applications that use this infrastructure.

The Junos OS Evolved infrastructure brings together multiple nodes into a single resource pool (that is. a distributed OS) wherein a Junos application can run on any node it is assigned to run on. This capability enables optimal resource utilization within an Junos Evolved system, and opens the door to new paradigms. Traditionally, the Routing Engines have x86 CPUs where Junos OS runs, and line cards, like FPCs or MPCs, have a general-purpose processor where the Junos Microkernel runs for Packet Forwarding Engine related functions. But in newer generations of x86 based line cards, a complete Linux operating system (OS) instance can be executed. Such line cards offer a more flexible framework and are capable of running processes and applications.

How the CLI in Junos OS Evolved Differs from Junos OS

In many ways, Junos OS Evolved is the same as Junos OS: Key applications such as the routing, bridging, and management software is the same in both. And management plane interfaces and APIs, such as CLI, NETCONF, JET, JTI, AFI, and underlying data models, remain highly consistent. There are, however, some differences in behavior. These differences are indicated throughout the Junos OS documentation. However, this section outlines the differences in one place, for your convenience.

- New CLI Statements and Commands (Junos OS Evolved) on page 11
- Modified CLI Statements and Commands (Junos OS Evolved) on page 13
- Changed CLI Command Output (Junos OS Evolved) on page 16
- Removed CLI Statements and Commands (Junos OS Evolved) on page 18

New CLI Statements and Commands (Junos OS Evolved)

The changes in infrastructure between Junos OS and Junos OS Evolved sometimes require different CLI configuration statements and operational commands. For example, there is a new hierarchy level of statements in Junos OS Evolved that are not in Junos OS: **[edit security host-vpn]**. For more on these new statements and commands, see Table 1 on page 11.

Table 1: New CLI Statements and Commands (Junos OS Evolved)

Statement or Command	Description	Link
New Statements		
[edit security host-vpn]	Configure a host-to-host VPN type of IPsec connection. Use the **connections**, **ike-log**, and **ike-secrets** statements at the **[edit security host-vpn]** hierarchy level to configure IKE and IPsec values.	*host-vpn*

Table 1: New CLI Statements and Commands (Junos OS Evolved) (continued)

Statement or Command	Description	Link
[edit security host-vpn connections]	You can configure the additional algorithms **aes256-sha384-modp3072** and **aes256-gcm128-modp3072** at each of the following hierarchy levels: • **[edit security host-vpn connections** *parent-connection-name* **ike-proposal]** • **[edit security host-vpn connections** *parent-connection-name* **children** *child-connection-name* **esp-proposal]**	*connections (Host VPN)* and *children*
[edit security host-vpn connections children *child-name***]**	Statements at this hierarchy level include **local-traffic-selector**, **remote**, and **remote-traffic-selector**.	*children*
[edit security host-vpn connections dpd-delay]	Statement to support dead peer detection. The dead peer detection delay sends keepalives to know if a peer has gone dead.	*connections (Host VPN)*
[edit security host-vpn ike-log]	Statements at the **[edit security host-vpn]** hierarchy level used to configure IKE and IPsec values.	*ike-log*
[edit security host-vpn ike-secrets]	Statements at the **[edit security host-vpn]** hierarchy level used to configure IKE and IPsec values.	*ike-secrets*
[edit security host-vpn remote]	Configure identity details for authenticating the remote device during IKE negotiations.	*remote (Host VPN)*
New Commands		
clear security host-vpn security-associations	Clear host IPsec security association information. You can configure host IPsec with the **[edit security host-vpn** statement.	*clear security host-vpn security-associations*
clear trace	Clear traces on. The trace data from all nodes is collected on the Routing Engine in **/var/log/traces**.	*clear trace*
request system application	Start a specific application on the node you specify.	*request system application*
request system debug-info	Collect debug information from Junos OS Evolved, such as logs. The logs are stored in the **/var/tmp/debug_collector_*timestamp*** directory. A **node** option is provided, allowing for information from a specific node to be collected.	*request system debug-info*
request system shutdown	Halt, power off, or reboot the entire system.	*request system shutdown (halt | power-off | reboot)*
show security host-vpn security-associations	Display host IPsec security association information for a specific security association or for all connections. You can configure host IPsec with the **host-vpn** statement at the **[edit security]** hierarchy level.	*show security host-vpn security-associations*
show security host-vpn version	Display the version of IPsec being used in the system.	*show security host-vpn version*

Table 1: New CLI Statements and Commands (Junos OS Evolved) (continued)

Statement or Command	Description	Link
show system applications	Display information about active applications on the system.	*show system applications*
show system errors	Display information about faults in the system. NOTE: For Junos OS Evolved, only the QFX5200 supports this command. For all other Junos OS Evolved platforms, use the *show system errors active*, *show system errors count*, *show system errors error-id*, or *show system errors fru* command.	*show system errors*
show system errors history	Display information about faults in the system that have been cleared. NOTE: For Junos OS Evolved, only the QFX5200 supports this command. For all other Junos OS Evolved platforms, use the *show system errors active*, *show system errors count*, *show system errors error-id*, or *show system errors fru* command.	*show system errors history*
show system software list	Display the installed versions on the Routing Engines in the system.	*show system software list*
show trace	Show the trace data from all nodes that are collected on the Routing Engine in **/var/log/traces**. You can refine how the trace data is shown by specifying trace time elapsed, application, process ID, and node. When disk usage of **/var/log/traces** exceeds the 5.7-GB limit, older trace files of applications are deleted to make space for new trace messages.	*show trace*

Modified CLI Statements and Commands (Junos OS Evolved)

Some CLI statements and commands in Junos OS Evolved have a different set of options from Junos OS. For a list of these changes, see Table 2 on page 13.

NOTE: For the CLI commands that produce changed output, see Table 3 on page 16.

Table 2: Modified CLI Statements and Commands (Junos OS Evolved)

Statement or Command	Change in Junos OS Evolved	Link
Modified Statements		
[edit interfaces *interface-name* unit *logical-unit-number* vlan-id]	The option **vlan-id 0** is not supported for this statement.	*vlan-id (VLAN ID to Be Bound to a Logical Interface)*
[edit policy-options policy-statement policy-name term *term-name* then]	The following options are not supported in Junos OS Evolved 18.3R1: **destination-class** and **source-class**.	*Actions in Routing Policy Terms*

Table 2: Modified CLI Statements and Commands (Junos OS Evolved) (continued)

Statement or Command	Change in Junos OS Evolved	Link
[edit system host-name]	When you specify the **host-name** statement, the Routing Engine name where the configuration is committed is appended to the hostname you specify.	*host-name*
[edit system login password]	The **format** option for this statement is limited to the following options: **(md5 | sha256 | sha512)**.	*format (System Login)*
[edit system syslog host]	The syslog host is connected through the management interface. You cannot configure the **source-address** option if you configure the **management-interface** option.	*host (System)*
Modified Commands		
clear ipv6 neighbors	In Junos OS Evolved, issuing the **clear ipv6 neighbors** command clears the cache for IPv6 neighbors in a reachable state.	*clear ipv6 neighbors*
configure	The **dynamic** option of the **configure** command is deprecated. The **configure dynamic** command is used to configure routing policies and certain routing policy objects in a dynamic database at the **[edit dynamic]** hierarchy level, a level you enter only by using the **configure dynamic** command. Because the **configure dynamic** command is deprecated, you cannot configure objects in a dynamic database, and you cannot use the **dynamic-db** statement.	*configure*
ping	The following options of the **ping** command are deprecated: • **detail** • **logical-system** • **loose-source** • **mac-address** • **strict** • **strict-source** • **vpls**	*ping*

Table 2: Modified CLI Statements and Commands (Junos OS Evolved) (continued)

Statement or Command	Change in Junos OS Evolved	Link
request system software add	The following options of the **request system software add** command are deprecated: • **best-effort-load** • **delay-restart** • **no-copy** • **on-primary** • (**re0** | **re1**) • **set** • **unlink** • **validate** • **validate-on-host** • **validate-on-routing-engine**	*request system software add*
request system software rollback	The following options are added to the **request system software rollback** command: • **(no-validate | validate)** • **with-old-snapshot-config**	*request system software rollback*
request system storage cleanup	A new option, **force-deep**, is added that cleans up all user-generated files as well.	*request system storage cleanup*
show firewall	The **application lsp** option is introduced, which you use to display implicit policers that are published by rpd.	*show firewall*
show host	The **routing-instance mgmt_junos** option is introduced.	*show host*
show interfaces voq	The **source-fpc** option is decprecated. For the PTX10003 router, there are only logical FPCs. Therefore, the **source-fpc** option is not required for this **show interfaces voq** command.	*show interfaces voq*
show system connections	The following options of the **show system connections** command are deprecated: **extensive** and **show-routing-instance**. The **node** option is introduced.	*show system connections*
show system core-dumps	The **node** option is introduced. the core dump files generated on the nodes are stored in the **/var/core/** directory.	*show system core-dumps*

Table 2: Modified CLI Statements and Commands (Junos OS Evolved) (continued)

Statement or Command	Change in Junos OS Evolved	Link
telnet	The following options of the **telnet** command are deprecated: • **bypass-routing** • **interface** • **logical-system** • **no-resolve** • **source**	*telnet*
traceroute	The following options of the **traceroute** command are deprecated: • **logical-system** • **next-hop** • **port** • **propogate-ttl**	*traceroute*

Changed CLI Command Output (Junos OS Evolved)

For changes in output for Junos OS Evolved, see Table 3 on page 16.

Table 3: Changed Command Output (Junos OS Evolved)

Command	Description of Change in Output	Link
clear interfaces statistics	Clears not only LACP statistics but also the counters displayed in the **show lacp statistics interfaces** command.	–
ping	When pinging a nonexistent route, the display output of the **ping** command does not print the number of packets sent or received or the number of packets loss.	*ping*
request system software delete	Output displays the version instead of the package.	*request system software delete*
request system software rollback	Output displays the version instead of the package.	*request system software rollback*
show chassis environment fpc	Displays different output.	*show chassis environment fpc*
show chassis fabric topology	Output displays **UP** for the State field, which should be **OK**.	–
show chassis fpc	The ouput displays only non-empty slots and the following modified slot states: **Unknown**, **Onlining**, **Offlining**, **Standby**, **Fault**, **Fault-off**, and **Spare**.	*show chassis fpc*

Table 3: Changed Command Output (Junos OS Evolved) (continued)

Command	Description of Change in Output	Link
show chassis fpc	The **show chassis fpc** command does not show any CPU or memory utililzation information because the PTX10003 device is a fixed-configuration router and the FPCs in it do not have dedicated CPUs.	*show chassis fpc*
show chassis pic status	The ouput displays only non-empty slots and the following modified slot states: **Unknown**, **Onlining**, **Offlining**, **Standby**, **Fault**, **Fault-off**, and **Spare**.	*show chassis pic*
show interfaces	LACP packets on the members of an AE interface are not counted as part of the Bundle Input Statistics in the **show interfaces ae *number* extesnsive** command output.	*show interfaces (Aggregated Ethernet)*
show interfaces	Configuration of IPv6 over the re0:mgmt-* interfaces is supported. However, the **show interfaces** command for re0:mgmt-*interfaces does not display the system-generated link-local address when IPv6 is configured over these interfaces.	NA
show interfaces detail	Output displays the **Last Flapped** field with the value **Never** after a Routing Engine reboot. The **Last Flapped** field provides details of the date, time, and how long ago the interface went up. The value **Never** signifies that the interface never flapped.	*show interfaces detail*
show interfaces extensive	Output does not display the Packet Forwarding Engine configuration and CoS default bandwidth allocation information.	*show interfaces (M Series, MX Series, T Series Routers, and PTX Series Management and Internal Ethernet)*
show multicast route extensive	Output displays the **Sensor ID** field that corresponds to a multicast route.	*show multicast route*
show multicast usage	Output displays the **Sensor ID** field that corresponds to a multicast route.	*show multicast usage*
show policer	Output displays only ARP policers. Use the **show application lsp** command to display implicit policers published by rpd.	*show policer* *show firewall*
show system uptime	Output displays only the **System booted** and **System-wide users** information. The output does not display information on current time, system booted, protocols started, or last configured parameters.	*show system uptime*
show system statistics arp	After running ping on an unreachable host, output shows that counts for **ARP requests received** and for **datagrams for an address no on the interface** are incremented.	–
show system statistics tcp	Output for the **show system statistics tcp** command is trimmed to show only fields supported in Junos OS Evolved.	*show system statistics tcp*

Table 3: Changed Command Output (Junos OS Evolved) (continued)

Command	Description of Change in Output	Link
show task replication	Output displays the same state whether the command is run from the master or spare Routing Engine.	*show task replication*
show version	Output of the **show version** command is changed to clearly show which Junos architecture is running on the device. Output of the **show version node all** command is revised to explicitly identify the Routing Engine in both the XML and CLI output.	*show version*

Removed CLI Statements and Commands (Junos OS Evolved)

For a listing of which CLI statements and commands are removed from Junos OS Evolved, see Table 4 on page 18. Where there is an alternative statement or command to use, it is noted in the table.

Table 4: Removed CLI Statements and Commands (Junos OS Evolved)

Statement or Command	Description
Removed Statements	
[edit chassis fpc *slot* error (fatal | major | minor) action offline-pic]	Deprecated.
traceoptions	The **traceoptions** option is removed at all hierarchy levels, except for the **[edit protocols]** hierarchy level. Starting in Junos OS Evolved Release 18.3R1, trace data from all applications on all nodes is collected on the Routing Engine in **/var/log/traces**. You can view collected traces with the **show trace** command. You can remove inactive tracing sessions with the **clear trace** command. NOTE: For Junos OS Evolved Release 18.3R1, the tracing feature is not supported for the **[edit protocols]** hierarchy level. For routing protocols, use the **traceoptions** option.
Removed Commands	
load override	Deprecated. Instead, you can use the **load update** command, which compares the candidate configuration and the new configuration data, and only changes the parts of the candidate configuration that are different from the new configuration.
request system (halt | power-off | power-on | reboot)	Deprecated. To manage the entire system, use the **request system shutdown** command.
request system software abort	Deprecated. There is no alternate command replacing it. The **request system software add** command has a built-in feature not to start an upgrade if a reboot is pending after an upgrade or rollback.
request system software (add | delete) set	Deprecated.

Table 4: Removed CLI Statements and Commands (Junos OS Evolved) (continued)

Statement or Command	Description
request system software in-service-upgrade	Deprecated. Use the **request system software add restart** command for ISSU. The **request system software add** command has a built-in feature not to start upgrade if a reboot is pending after an upgrade or rollback.
request system software set	Deprecated. To set the current system to an installed software version, use the **request system software rollback reboot** command.
request system snapshot	Deprecated; you can no longer make system snapshots. To upgrade the system, use the **request system software rollback** command. To see what releases of the OS are on the Routing Engines, use the **show system software list** command.
set date	If you issue the **set date** command, an error occurs. The system uses NTP, which is always running, to get date information.
show chassis fabric unreachability	Deprecated. See the **show system errors** command for similar functionality.
show chassis fabric summary	The **show chassis fabric summary** command is removed. See the **show system errors** command for similar functionality.
show chassis network-services	Deprecated.
show class-of-service forwarding-table	Deprecated. The removed options include **classifier**, **classifier mapping**, **drop-profile**, **policer**, **rewrite-rule**, **rewrite-rule mapping**, **scheduler-map**, and **shaper**.
show database-replication	Deprecated.
show interfaces em0 | em1	The em0 and em1 Ethernet management interfaces are removed. Use re0:mgmt-* for Routing Engine 0 (Routing Engine 1 would be re1:mgmt-*).
show interfaces ixgbe0 | ixgbe1	The ixgbe0 and ixgbe1 internal interfaces are removed.
show interfaces mac-database	Deprecated. The **MAC accounting and policing not supported** message is displayed.
show pfe	Deprecated.
show system switchover	Deprecated.

Where to Find Information on Common Procedures

This guide, *Introducing Junos OS Evolved,* has information about the features and changes in the next generation of Junos OS. However, much about using Junos OS remains the same. Junos OS Evolved has the same CLI user interface, some of the same processes, and the same management and automation tools as Junos OS. You configure and manage Junos OS Evolved the same way as you always have configured and managed Junos OS.

For your convenience, this section lists some links to the Junos OS documentation you might want to consult.

- *Initial Router or Switch Configuration Using Junos OS*—Overview of initial configuration.
- *CLI User Guide*—Procedures on configuring and managing the CLI.
- *Getting Started Guide*—More procedures for initial configuration.
- *User Access and Authentication Feature Guide*—Procedures on granting access and setting up authentication on your device.
- *Network Management and Monitoring Guide*—Procedures on SNMP, remote monitoring (RMON), destination class usage (DCU) and source class usage (SCU) data, accounting profiles, and logging.
- *Installing the Software Package on a Router with a Single Routing Engine*—Procedure on installing Junos OS on a device with one Routing Engine.
- *Junos OS Installation Packages Prefixes*—Overview of install packages by prefix, including Junos OS Evolved images.

CHAPTER 2

Booting Junos OS Evolved from a USB

- Booting Junos OS Evolved by Using a Bootable USB Drive on page 21

Booting Junos OS Evolved by Using a Bootable USB Drive

You can boot Junos OS Evolved from a USB device. Booting from the USB device reformats the disk and reinstalls the software without prompting you. After the installation is done, the device waits for the USB drive to be removed from the USB port and then reboots into the new version.

There are several ways to create the Junos OS Evolved image on the USB drive. Also included are a procedure for booting from the USB drive and one for how to recover if the boot from the USB goes bad.

- Create a Bootable USB Drive Using a Windows Device on page 21
- Create a Bootable USB Drive Using a MAC OS X on page 22
- Create a Bootable USB Drive Using a Switch or Router Running Junos OS Evolved on page 23
- Boot Junos OS Evolved from a Bootable USB Drive on page 24
- Recover Junos OS Evolved Using USB Scratch Install on page 25

Create a Bootable USB Drive Using a Windows Device

You need the following items to perform this procedure:

- Windows desktop or laptop with a USB port.
- Version 2.0 or version 3.0 USB device with the following features:
  - USB device is big enough to hold the ISO image.
  - USB device must have no security features, such as a keyed boot partition.
- Junos OS Evolved ISO image

  For a virtual Windows desktop you must map a physical USB of the host to the guest virtual machine (VM).

To create a bootable USB drive using a Windows device:

1. Install Win32 Disk Imager on your laptop or computer.

   You can download it from https://sourceforge.net/projects/win32diskimager/.

2. Download the required Junos OS image from the Downloads page to the Documents directory of your laptop or computer.

3. Insert a USB flash drive into the USB port of your laptop or computer.

4. Open the win32diskimager application and, in the **Image File** box, type the path to the Documents directory (or click the folder icon to navigate to the Documents directory) and select the install media image.

5. Under **Device**, select the USB flash-drive and click **Write and Confirm**. The Progress box shows the progress.

6. Remove the USB flash drive once it is complete.

   The USB flash-drive is now ready to use as a bootable disk.

Create a Bootable USB Drive Using a MAC OS X

You need the following items to perform this procedure:

- A MAC OS X desktop or laptop with a USB port.
- Version 2.0 or version 3.0 USB device with following features:
  - USB device is big enough to hold the ISO image.

To create a bootable USB using MAC OS X:

1. Copy the install media (.img format) to the **/var/tmp/** directory of the routing device using the **scp** command.

   For example:

   $ scp *user*@*server*:/var/tmp/*image-name* /var/tmp/

   ```
   password:
   ```

2. To get the list of devices on the MAC OS X device, run the **diskutil list** command.

3. Insert the USB flash drive into the USB port of the MAC OS X.

4. Run the **diskutil list** command again to determine the device node assigned to USB flash-drive (for example, **/dev/disk3**).

5. Run the **diskutil unmountDisk /dev/disk*N*** command.

   Replace ***N*** with the disk number from the last command. (In this example, ***N*** would be 3.)

   For example:

   ```
   $ diskutil unmountDisk /dev/disk3
   Unmount of all volumes on disk3 was successful
   ```

6. Execute the command **sudo dd if=/var/tmp/junos-install-media-usb-srxhe-x86-64-17.4R1-S1.9.img of=/dev/rdiskN bs=1m**

   For example:

   ```
   $ sudo dd if=/var/tmp/usb.img of=/dev/rdisk3 bs=1m
   Password:
   965+0 records in
   965+0 records out
   1011875840 bytes transferred in 82.891882 secs (12207177 bytes/sec)
   ```

7. The USB with image is created and ready for installation. Safely remove the USB drive and use it as a bootable USB drive on the device on which you plan to run Junos OS Evolved.

Create a Bootable USB Drive Using a Switch or Router Running Junos OS Evolved

You need the following items to perform this procedure:

- A switch or router with a USB port that is running Junos OS Evolved.
- Version 2.0 or version 3.0 USB device with following features:
  - USB device is big enough to hold the ISO image.
  - USB device must have no security features, such as a keyed boot partition.
  - USB device label should be **JUNOS**.

To create a bootable USB using a switch or router running Junos OS Evolved:

1. Download **.img** image from Downloads site and copy it to the **/var/tmp/** directory of the switch or router running Junos OS Evolved using the **scp** command.

2. Enter the shell as root:

   ```
   user@switch> start shell user root
   Password:
   ```

3. Before inserting the USB device, list the contents of **/dev/**.

   ```
   root@re0:~#ls /dev/sd*
   ```



```
/dev/sda   /dev/sda3  /dev/sda6  /dev/sdb1  /dev/sdb4  /dev/sdb7
/dev/sda1  /dev/sda4  /dev/sda7  /dev/sdb2  /dev/sdb5
/dev/sda2  /dev/sda5  /dev/sdb   /dev/sdb3  /dev/sdb6
root@re0:~#
```

4. Insert the USB drive in the USB port.

5. Repeat the command to list the contents of **/dev/**.

```
root@re0:~#ls /dev/sd*
/dev/sda   /dev/sda3  /dev/sda6  /dev/sdb1  /dev/sdb4  /dev/sdb7
/dev/sda1  /dev/sda4  /dev/sda7  /dev/sdb2  /dev/sdb5  /dev/sdc
/dev/sda2  /dev/sda5  /dev/sdb   /dev/sdb3  /dev/sdb6  /dev/sdc1
root@re0:~#
```

NOTE: **/dev/sdc** is the USB drive.

6. Execute the following command, where **$USB** identifies the device for that USB (typically **sdc** in Linux):

```
dd if=/var/tmp/usb.img of=/dev/$USB bs=100000
```

7. The USB with image is created and ready for installation. Safely remove the USB drive and use it as a bootable USB drive on the device on which you plan to run Junos OS Evolved.

Boot Junos OS Evolved from a Bootable USB Drive

To perform this procedure, you must first create a USB drive with the Junos OS Evolved software image installed on it. For instructions, see "Create a Bootable USB Drive Using a MAC OS X" on page 22 or "Create a Bootable USB Drive Using a Switch or Router Running Junos OS Evolved" on page 23.

To install Junos OS Evolved on a device that runs Junos OS Evolved using a USB drive:

1. Connect to the console.

2. Insert the USB drive with the Junos OS Evolved package in the **USB0** port on the routing device.

3. Reboot the routing device from the CLI:

```
user@host> request system shutdown reboot usb
```

4. Remove the USB device.

Junos OS Evolved automatically installs.

Recover Junos OS Evolved Using USB Scratch Install

Problem **Description:** If, while you are trying to boot Junos OS Evolved from a USB device, the device goes to a bad state, follow this procedure.

Solution To recover using a USB scratch install:

1. Insert the bootable USB device into the device.
2. Access the BIOS manager to check the USB selection:
   a. Reboot the routing device.

      ```
      user@host> request system shutdown reboot usb
      ```

   b. To access the BIOS boot manager, press ESC while the system reboots.
3. In the BIOS boot manager, select one of the following:
   - For PTX10003 devices, select **EFI USB**.
   - For QFX5200 devices, select **USB: *model-name***.

   The scratch installation starts automatically and the operating system is installed.
4. Remove the USB or type a reboot command.

CHAPTER 3

Running Third-Party Applications on Junos OS Evolved

- Running Third-Party Applications in Containers on page 27
- Running Linux Tools on Junos OS Evolved on page 28

Running Third-Party Applications in Containers

There are two types of containers in which you can run third-party applications: LXC containers and docker containers.

- Deploying an LXC Container on page 27
- Deploying a Docker Container on page 28

Deploying an LXC Container

To deploy a supported version of a linux container, either the download container or use a container you already have. In either case, you use the standard **lxc-create** command with the **--template** argument. The **--template** argument can point to a local image or download an image. If the image is local, only the **--name** and **--template** arguments are required. Non-local images might require the **--server** argument and the **-d**(distribution), **-r**(release), and **-a**(architecture) arguments.

To deploy an LXC container:

1. Start the LXC service:

   ```
   [vrf:none] user@host_RE0:~# systemctl start lxc
   ```

2. Switch to vrf0 to get network access in order to pull the image:

   ```
   [vrf:none] user@host_RE0:~# switchvrf $$ vrf0
   ```

3. Create the LXC container.

   In the following example, you are downloading an pyez container from a server:

   ```
   [vrf:vrf0] user@host_RE0:~# /usr/bin/lxc-create --name pyez1 --template download --
   --server 10.48.19.245/lxc-images/ --no-validate -d pyez_new -r 2.1.9 -a amd64
   ```

Deploying a Docker Container

To deploy a docker container:

1. Start the docker service using the vrf0 socket:

   ```
   [vrf:vrf0] user@host_RE0:~# systemctl start docker@vrf0
   ```

2. Import the image.

   In the following example, you import the same pyez image as was used for the LXC container deployment procedure.

   NOTE:

   The URL for the import command would change for different containers.

   ```
   [vrf:vrf0] user@host_RE0:~# docker -H unix:///run/docker-vrf0.sock import
   http://10.48.19.245/lxc-images/images/pyez_new/2.1.9/amd64/default/20190225_19:53/rootfs.tar.xz
   ```

3. Make sure the image is downloaded, and get the image ID.

   ```
   [vrf:vrf0] user@host_RE0:~# docker -H unix:///run/docker-vrf0.sock image ls
   REPOSITORY          TAG                 IMAGE ID            CREATED
      SIZE
   <none>              <none>              738c70533604        59 seconds ago
      491MB
   ```

4. Create a container using the image ID and enter a bash session in that container.

   ```
   [vrf:vrf0] user@host_RE0:~# docker -H unix:///run/docker-vrf0.sock create -it --name
   pyez1 --network=host 738c70533604 bash
   ```

Running Linux Tools on Junos OS Evolved

Junos OS Evolved is Junos OS except that it runs on native Linux and, therefore, can accommodate your running third-party applications. There are some differences between the way Linux displays requested network topology information such as interface and route data and the way Junos OS displays this information. The CLI is designed to overcome these differences. But typically, third-party applications running on native Linux obtain this information directly from the native Linux sources using shell commands.

Junos OS Evolved comes equipped with an intercept mechanism that redirects shell requests for network topology information away from the Linux stack to a space where the information can be obtained from Junos OS. This intercept mechanism is accomplished by way of an intercept library, libnli.so, that you preload. After you preload

the intercept library, certain types of requests are intercepted and show Junos OS information.

- Example of a Preloaded Linux Command on page 29
- Interface Name Translation on page 32
- Other Caveats for the Intercept Feature on page 34

Example of a Preloaded Linux Command

An example how the the preload directive works follows using the command **ifconfig**, which displays interfaces.

If you preload the **ifconfig** command with the intercept library, Junos OS interface information is returned. Notice that the intercept library only translates logical interfaces. In this example, because there are logical interfaces only on lo0 and re0:mgmt-0.0, the output displays only these two interfaces for the preloaded **ifconfig** command.

```
[vrf:none] user@host_RE0:~# LD_PRELOAD=libnli.so ifconfig
lo0_0     Link encap:Ethernet  HWaddr 00:00:00:00:00:00
          inet addr:128.102.224.244  Mask:255.255.255.255
          inet6 addr: abcd::128:102:224:244/128 Scope:Global
          inet6 addr: fe80::5668:a6f0:6e:b79/128 Scope:Link
          UP LOOPBACK RUNNING  MTU:65535  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

mgmt-0-00-0000 Link encap:Ethernet  HWaddr 56:68:a6:6e:0b:79
          inet addr:10.102.224.244  Bcast:10.102.239.255  Mask:255.255.240.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1103938 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1905 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:85166899 (81.2 MiB)  TX bytes:243066 (237.3 KiB)
```

You can get the same results by running jbash, which is a shell provided with Junos OS Evolved that preloads libnli.so and libsi.so by default.

CAUTION: Only use jbash to get the network state information. Don't use jbash as your default shell.

If you issue the command without preloading it with the intercept library, the output shown is from Linux. Notice that the following output is longer than that from Junos OS. Linux does not make the distinction between physical interfaces and logical interfaces that the Junos CLI does.

```
[vrf:none] user@host_RE0:~# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 56:68:a6:6e:0b:79
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:1608443 errors:44 dropped:0 overruns:0 frame:44
```

```
          TX packets:2652 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:150837081 (143.8 MiB)  TX bytes:341675 (333.6 KiB)

eth1      Link encap:Ethernet  HWaddr 56:68:a6:6e:0b:7e
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:9600  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:418 (418.0 B)

eth2      Link encap:Ethernet  HWaddr 56:68:a6:6e:0b:83
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:9600  Metric:1
          RX packets:907046 errors:0 dropped:0 overruns:0 frame:0
          TX packets:926156 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:70342248 (67.0 MiB)  TX bytes:119965968 (114.4 MiB)

eth3      Link encap:Ethernet  HWaddr 56:68:a6:6e:0b:8d
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth4      Link encap:Ethernet  HWaddr 56:68:a6:6e:0b:9d
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:1607983 errors:44 dropped:0 overruns:0 frame:44
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:150335380 (143.3 MiB)  TX bytes:0 (0.0 B)

ingvrf    Link encap:Ethernet  HWaddr 12:6e:39:d6:5a:64
          UP RUNNING NOARP MASTER  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

iri       Link encap:Ethernet  HWaddr 4e:a2:93:c0:ac:67
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP RUNNING NOARP MASTER  MTU:65536  Metric:1
          RX packets:2199380 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2216726 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:674308465 (643.0 MiB)  TX bytes:735412009 (701.3 MiB)

jtd0      Link encap:Ethernet  HWaddr 06:50:4e:19:c6:c5
          inet6 addr: fe80::450:4eff:fe19:c6c5/64 Scope:Link
          UP BROADCAST RUNNING NOARP  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

jtdrop    Link encap:Ethernet  HWaddr ba:d0:d0:72:7e:eb
          inet6 addr: fe80::b8d0:d0ff:fe72:7eeb/64 Scope:Link
          UP BROADCAST RUNNING NOARP  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
```

```
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

jtdv0     Link encap:Ethernet  HWaddr 56:2a:0c:39:f1:5d
          inet6 addr: fe80::542a:cff:fe39:f15d/64 Scope:Link
          UP BROADCAST RUNNING NOARP  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:280 (280.0 B)

jtdv50    Link encap:Ethernet  HWaddr 56:5e:67:d6:e2:d2
          inet6 addr: fe80::545e:67ff:fed6:e2d2/64 Scope:Link
          UP BROADCAST RUNNING NOARP  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:280 (280.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:32 errors:0 dropped:0 overruns:0 frame:0
          TX packets:32 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:2144 (2.0 KiB)  TX bytes:2144 (2.0 KiB)

mgmt_junos Link encap:Ethernet  HWaddr 6a:75:4b:20:d0:4e
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP RUNNING NOARP MASTER  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

sit0     Link encap:UNSPEC  HWaddr 00-00-00-00-30-30-30-00-00-00-00-00-00-00-00-00

          NOARP  MTU:1480  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

tunl0     Link encap:IPIP Tunnel  HWaddr
          NOARP  MTU:1480  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

vcb       Link encap:Ethernet  HWaddr 56:68:a6:6e:0b:83
          inet addr:176.1.1.1  Bcast:0.0.0.0  Mask:255.255.255.252
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:9600  Metric:1
          RX packets:907043 errors:0 dropped:0 overruns:0 frame:0
          TX packets:924347 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:57643466 (54.9 MiB)  TX bytes:118743890 (113.2 MiB)
```

```
vfb       Link encap:Ethernet  HWaddr 56:68:a6:6e:0b:7e
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:9600  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

vib       Link encap:Ethernet  HWaddr 3e:fb:67:87:16:1a
          inet addr:128.0.0.4  Bcast:0.0.0.0  Mask:255.0.0.0
          inet6 addr: fe80::3cfb:67ff:fe87:161a/64 Scope:Link
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:74 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:3420 (3.3 KiB)

vmb0      Link encap:Ethernet  HWaddr 56:68:a6:6e:0b:79
          inet addr:10.102.224.244  Bcast:0.0.0.0  Mask:255.255.240.0
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:1602504 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2645 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:124666750 (118.8 MiB)  TX bytes:340201 (332.2 KiB)

vmb1      Link encap:Ethernet  HWaddr 56:68:a6:6e:0b:9d
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:1602784 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:124008554 (118.2 MiB)  TX bytes:0 (0.0 B)

vrf0      Link encap:Ethernet  HWaddr ca:12:9e:40:a8:01
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP RUNNING NOARP MASTER  MTU:65536  Metric:1
          RX packets:124413 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2597 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:19087613 (18.2 MiB)  TX bytes:338185 (330.2 KiB)

vrf50     Link encap:Ethernet  HWaddr 06:de:d7:3d:18:be
          UP RUNNING NOARP MASTER  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Interface Name Translation

One limiting factor to using this intercept mechanism is that Linux interface naming is incompatible with the Junos OS interface naming. Linux supports 15-byte interface names (15 + null-character); network interface names that exceed this limit are truncated in outputs. Junos OS logical interface names could be longer than 15 bytes, for example, et-0/0/10:2.32767.

To work around this difference, Junos OS Evolved uses a translation rule (see Table 5 on page 33) to render logical interface names in a Linux-compliant format. The translation renders a format such as name-fpcSlot/picSlot/port:channelId.subUnit to nn-ffpttccssss. Using interface names translated according to this rule, third-party applications can effectively fetch the topology information from Junos OS.

Only translation of logical interface names is supported, and translation of both channelized and nonchannelized logical interface names is supported.

Table 5: Translation Rule for Interface Names

Value	Description	Allotted Space (in bytes)	Range
nn	mapped name bytes	2	
ff	fpc in hex	2	0-255
p	pic in hex	1	0-15
tt	port number in hex	2	0-255
cc	channel in hex; use “xx” if not present	2	0-255
ssss	subunit in hex	4	0-65535
Except for management interfaces, if the logical interface name does not have a hyphen (-) in it, the dot (.) in the name is changed to an underscore (_), for example: ifdname.subunit gets translated to ifdname_subunit.			
For management interfaces, reX:mgmt-Y.Z translates to mgmt-x-yy-zzzz, where x, yy, zzzz are in hex-padded with 0 for a fixed length. And the reverse translation happens on the same lines.			

See Table 6 on page 33 for examples of Junos logical interface names and their Linux-compliant forms.

Table 6: Examples of Translated Logical Interface Names

Junos Logical Interface Name	Translated Linux-Compliant Interface Name
et-1/2/3.4	et-01203xx0004
ge-1/2/3.32	ge-01203xx0020
et-1/15/3.4	et-01f03xx0004
et-1/2/255:6.7	et-012ff060007
et-1/2/4:5.32767	et-01204057fff
re0:mgmt-1.2	mgmt-0-01-0002

Table 6: Examples of Translated Logical Interface Names (continued)

Junos Logical Interface Name	Translated Linux-Compliant Interface Name
ae0.1	ae0_1
irb0.11	irb0_11

When accessing Junos OS states by preloading libnli.so, the interface name in the output is shown as a translated Linux-compliant interface name. You must also use the translated Linux-compliant interface name when using it as an argument in a command. The translated et-01000000000 interface name is used as an argument in the following example:

```
[vrf:none] user@host_RE0:~# LD_PRELOAD=libnli.so ifconfig et-01000000000
et-01000000000 Link encap:Ethernet  HWaddr 5c:31:b0:35:01:ff
          inet addr:20.20.20.24  Bcast:20.20.20.255  Mask:255.255.255.0
          inet6 addr: 2000:200:20::2/64 Scope:Global
          inet6 addr: fe80::5e31:b0ff:fe35:1ff/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1514  Metric:1
          RX packets:312 errors:0 dropped:0 overruns:0 frame:0
          TX packets:156 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:31004 (30.2 KiB)  TX bytes:21346 (20.8 KiB)
```

Other Caveats for the Intercept Feature

This intercept feature supports read-only requests. Any write request returns an error.

Representation of certain Junos network state may not be mappable to Linux equivalents. In these cases, the data is either be omitted or re-mapped to a comparable Linux model. For example, Junos OS Evolved supports a rich suite of nexthop types such as **composite** or **unilist** that do not have comparable implementations in native Linux.

Third-party applications that are linked statically cannot be intercepted and, therefore, are not supported by this feature.