

Introducing Junos OS Evolved

Published
2023-03-14

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

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

Introducing Junos OS Evolved

Copyright © 2023 Juniper Networks, Inc. All rights reserved.

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

About This Guide | vii

1

Overview of Junos OS Evolved

Junos OS Evolved Overview | 2

Top Differences Between Junos OS Evolved and Junos OS | 4

How Junos OS Evolved Differs from Junos OS | 13

Default Directories for Junos OS Evolved File Storage | 56

Junos OS Evolved Components and Processes | 58

Error TPAs for Route Installation | 61

Overview of Error Third-Party Attachments (TPAs) on Errored Objects During Route Installations | 61

Set Up the System for Error TPAs | 62

CLI Commands for Viewing Error Details | 63

Shell Commands for Junos OS Evolved | 64

Where to Find Information on Common Procedures | 66

2

Junos OS Evolved Configuration Overview

Junos OS Evolved Configuration Basics | 68

Methods for Configuring Junos OS Evolved | 68

Junos OS Evolved Configuration from External Devices | 71

3

Running 3rd Party Applications with Junos OS Evolved

Overview of Third-Party Applications on Junos OS Evolved | 73

Introduction to Third-Party Applications on Junos OS Evolved | 73

Running Applications in Containers vs Using Signing Keys | 73

Applications in Containers | 74

Using Signing Keys | 74

Application Pre-requisites | 74

Application APIs | 75

Security Caveats | 76

File Security with IMA | 76

Managing Third-Party Applications | 77

How to Run Applications | 77

Using Intercept Libraries | 78

Example of a Preloaded Linux Command | 79

Interface Name Translation | 84

Caveats for the Intercept Feature | 86

Removing Third-Party Applications | 87

Running Third-Party Applications in Containers | 88

Deploying a Docker Container | 89

Managing a Docker Container | 90

Enabling Netlink or Packet IO in a Container | 90

Selecting a VRF for a Docker Container | 91

Modifying Resource Limits for Containers | 92

Signing Third-Party Applications to Run Natively on Junos OS Evolved | 93

Signing Keys Overview | 93

Generating Signing Keys | 94

Generating Signing Keys Using the OpenSSL Command-Line | 94

Generating Signing Keys Using an OpenSSL Configuration File | 95

Importing Signing Keys into the System Keystore and IMA Extended Keyring | 96

Viewing the System Keystore and IMA Extended Keyring | 97

How to Sign Applications | 98

How to Run Signed Applications | 99

Building Third-Party Applications | 99

JET SDK for Junos OS Evolved | 100

- Downloading the JET SDK | 100
- Installing the JET SDK for Junos OS Evolved | 101

SysMan and systemd Controlled Applications | 101

Folder Structure for Third-Party Applications | 102

Third-Party Application Files | 103

- Makefile | 103
- SRC Files | 103
- Script Files | 103
- Service Files | 104
- .yaml Files | 105

Creating a Third-Party Package | 107

- Create a SysMan Managed Package | 107
- Create a systemd Managed Package | 107
- Jet-evo Tool Configuration | 108
- Installing a Third-Party Package | 109

4

Finding Software Documentation for Junos OS Evolved

Where to Find Software Documentation for Junos OS Evolved | 114

Authentication and Access Control | 115

Chassis Features | 116

Class of Service | 118

Ethernet Switching and Bridging | 120

Forwarding Options | 123

High Availability | 124

Interfaces | 126

IPv6 | 129

Multicast | 131

Multiprotocol Label Switching Applications | 133

Network Management and Monitoring | 135

Network Security | 138

Routing Options | 140

Routing Policies and Firewall Filters | 142

Routing Protocols | 145

Services Applications | 148

System Basics | 150

System Management | 151

Virtual Private Networks | 154

5

Operational Commands

show nlsd mode | 159

show nsagentd (statistics | status) | 160

About This Guide

Use this guide to become acquainted with Junos OS Evolved, a unified, end-to-end network operating system. Learn about its strengths, similarities to, and differences from Junos OS.

1

CHAPTER

Overview of Junos OS Evolved

[Junos OS Evolved Overview | 2](#)

[Top Differences Between Junos OS Evolved and Junos OS | 4](#)

[How Junos OS Evolved Differs from Junos OS | 13](#)

[Default Directories for Junos OS Evolved File Storage | 56](#)

[Junos OS Evolved Components and Processes | 58](#)

[Error TPAs for Route Installation | 61](#)

[Shell Commands for Junos OS Evolved | 64](#)

[Where to Find Information on Common Procedures | 66](#)

Junos OS Evolved Overview

IN THIS SECTION

- [Benefits | 2](#)
- [Native Linux Base | 3](#)
- [Integrated Database for State | 3](#)
- [Modular Design | 4](#)

Junos OS Evolved is a unified, end-to-end network operating system that provides reliability, agility, and open programmability for successful cloud-scale deployments. With Junos OS Evolved, you can enable higher availability, accelerate your deployments, innovate more rapidly, and operate your network more efficiently. We've aligned Junos OS Evolved with Junos OS so that you can seamlessly continue to manage and to automate your network.

Benefits

Junos OS Evolved provides several benefits to Juniper Networks customers:

- It runs natively on Linux, providing direct access to all the Linux utilities and operations. With Linux integration, you can use standard Linux and open-source tools to speed up onboarding, accelerate feature adoption with a smooth upgrade process, and enjoy enhanced debugging capabilities for streamlined qualification and deployment.
- Support for 3rd party applications and tools. You can run Linux applications directly on Junos OS Evolved using Docker containers, or create custom applications for advanced networking solutions. You can use existing Linux tools and procedures to create custom functions on a developer-friendly platform with a short learning curve. This versatility allows you to create the solution that best fits your needs through simple third-party application integration and the ability to implement the components required for specific use cases.
- You can install multiple different Junos OS Evolved software releases on a device, with support for rolling back to previous versions. This gives you the flexibility to try out different software releases and easily revert back to your preferred version if necessary.

- Enhanced security at all OS layers. Junos OS Evolved uses an integrity solution called Integrity Measurement Architecture (IMA), and a companion mechanism called the Extended Verification Module (EVM). These open source protections are part of a set of Linux Security Modules that are industry-standard and consistent with the trust mechanisms specified by the Trusted Computing Group. Junos OS Evolved also supports other security features such as TPM infrastructure, hardened secure BIOS, and secure boot. Security is a core design principle for Junos OS Evolved. Juniper Networks is committed to maintaining a strong security infrastructure to keep your network safe and protected.
- Nearly all of the CLI and user interfaces are identical to those provided in Junos OS, meaning you can pick up Junos OS Evolved with a minimal learning curve. These similarities provide simplicity and operational consistency, minimizing the effort required to implement, maintain, and customize your end-to-end solution.

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 the processes run. Effectively, this decoupling 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.

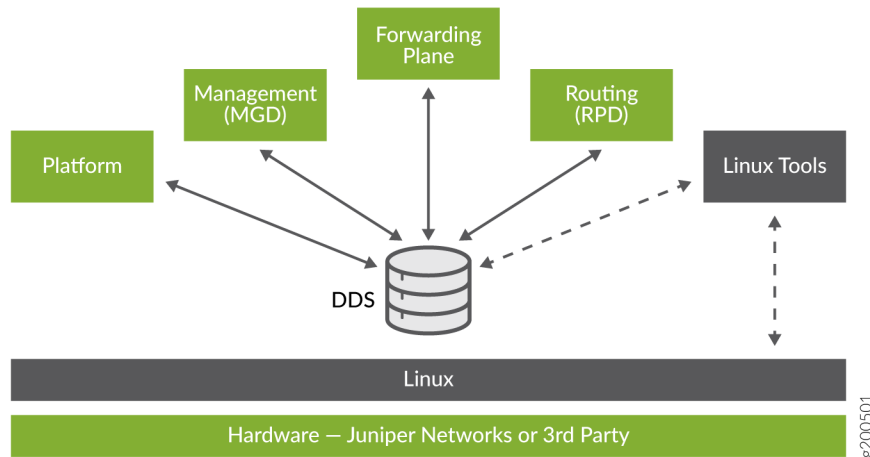
Integrated Database for State

State is the retained information or status about physical or logical entities that the system preserves and shares across the system, and supplies during restarts. State includes both operational and configuration state, including committed configuration, interface state, routes, and hardware state. In Junos OS Evolved, state can be held in a 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 4). 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. Applications can be individually restarted without requiring a system reboot. Restarted applications reload the state that is preserved in the DDS.

Top Differences Between Junos OS Evolved and Junos OS

IN THIS SECTION

● System Differences | 5

- Software Structure and Applications | 6
- State Model | 7
- Software Management | 7
- Management Interfaces | 8
- Routing Engine Firewall Filters | 9
- Junos OS Evolved Network Stack | 10
- System Logging | 11
- System Log Message Format | 11
- Tracing Architecture | 11

Although we've aligned Junos OS Evolved with Junos OS, there are some key differences to keep in mind when operating Junos OS Evolved. Junos OS Evolved is built on top of a Linux kernel, while Junos OS operates on the FreeBSD kernel. This and other fundamental differences in the design of Junos OS Evolved may be relevant in the management of your network. Read on to learn about the top differences between Junos OS Evolved and Junos OS.

System Differences

The concept of *system* in Junos OS Evolved is different from Junos OS. Junos OS uses a Routing Engine centric model, where *system* usually refers to a Routing Engine. However, Junos OS Evolved uses a *node*-based model, where *system* refers to all nodes, including Routing Engines, Flexible PIC Concentrators (FPCs), and more. In Junos OS Evolved, a *node* is any component that can run the Linux kernel and Junos OS Evolved applications, and all nodes are considered compute nodes.

Operational Impact

In Junos OS Evolved you can perform many actions on a per-node basis. You can use CLI commands to view information and request operations on individual nodes.

Relevant CLI Commands

- `show system nodes` — View a list of all nodes in the system.
- `show node (reboot | statistics) node-name` — View information about a specific node.
- `show system applications <node node-name>` — Display application summary information for all nodes or a specific node.

- `show system core-dumps <node node-name>` — Show system core files for all nodes or a specific node.
- `show system errors active`—Use this command instead of the `show chassis errors active` command to view system error information.
- `show system processes <node node-name> <detail>` — Display process information for all nodes or a specific node.
- `show system storage node (re0 | re1 | fpc0 | fpc1 | ...)` — View the free disk space for a specific node.
- `show version node (all | node-name)` — Display software version information for all nodes or a specific node.
- `request node (halt | offline | online | power-off/on | reboot) node-name` — Request an operation on a specific node.
- `request system reboot` — In Junos OS Evolved this command will reboot all nodes.

Software Structure and Applications

Junos OS Evolved functions as a distributed Linux OS with processes running as self-contained applications. Every Junos OS Evolved process runs as an application. All Junos OS Evolved applications are managed by the `systemd` process using service units. Applications run as separate services, which provides fault isolation because you can restart an application separately without impacting other applications. Most applications publish and consume state, which is stored in a central database.

Operational Impact

In Junos OS Evolved, many high availability features are per-application rather than per-node. Some applications run a full-time backup for rapid failover, while other applications are restarted on a new node in the event of a failure.

Relevant CLI Commands

- `show system applications <node node-name>` — Display application summary information for all nodes or a specific node.
- `restart process` — In Junos OS this command restarts a specific process. In Junos OS Evolved the same command restarts a specific application (process) on the same node from which the command is issued.
- `request system application restart app application node node` — This command is specific to Junos OS Evolved and restarts a specific application on a specific node.

State Model

Junos OS Evolved uses a distributed state infrastructure. Applications publish or subscribe to state objects, which are stored in a state database called the Distributed Data Store (DDS) that is distributed across nodes. By comparison, Junos OS processes store state internally, exchanging state information and state changes with other processes through the kernel. The Junos OS Evolved state model is asynchronous and eventually consistent at the transport layer with causal consistency at the application layer when accessing state. This means that if a process restarts in Junos OS Evolved, information is not lost because it can retrieve state information from the DDS.

Operational Impact

The Junos OS Evolved state model leads to faster performance because you don't have to wait for the slowest component to update. Applications read from and write to system state without waiting for every other process to first complete updates. If an application restarts, state is preserved and retrieved from the DDS by the new instance, even if the application is spawned on a different node.

Software Management

Each time you install a software image on Junos OS Evolved, the previous software image and configuration are preserved automatically. Junos OS Evolved stores software images in the `/soft` directory. Each version of the software is stored in a distinct area, thus ensuring that a software package installation does not affect the other software versions installed on the system. While Junos OS supports installing two software versions on the device, Junos OS Evolved supports storing as many software images as space allows. However, we recommend that you keep no more than five versions of software on the system.

During a successful installation, the installation package completely re-installs the existing software. It retains configuration files and similar information, such as secure shell and host keys, from the previous version. When you reboot the system after a software package installation, all the Routing Engines and FPCs in the system run the new version of the software.

Operational Impact

Junos OS Evolved ensures that all Routing Engines and FPCs in the system are running the same software version. When you install a software image on the primary Routing Engine, the system installs the new version of software on both Routing Engines, if the Routing Engines are online and part of the system. If you insert a Routing Engine that has a different software version into the system and you have not configured the `system auto-sw-sync enable` statement, the Routing Engine is kept outside the system, and the system generates a software mismatch alarm.

When you install a new software image, the previous software package is preserved in a separate area, and you can manually roll back to it if necessary. Junos OS Evolved enables you to roll back to an alternate image with either the current configuration file or with the configuration snapshot from when the alternate image was last running.

Relevant CLI Commands

- `show system software list` — On Junos OS Evolved, view the currently installed images on each node.
- `show system storage` — View available storage space. On Junos OS Evolved, the `/soft`, `/var`, and `/data` directories must have less than 90% capacity to install additional images.
- `request system software delete` — Clean up old images. Starting in Junos OS Evolved Release 20.1R1, use this command instead of the `request system storage cleanup` command to remove ISO images from the system.
- `request system snapshot` — Take a snapshot of the files currently used to run the device, and copy the files onto the alternate solid-state drive (SSD). The snapshot includes the complete contents of the `/soft`, `/config`, and `/root` directories, copies of user data, and content from the `/var` directory (except the `/var/core`, `/var/external`, `/var/log`, and `/var/tmp` directories).
- `request system software rollback reboot <package-name> <with-old-snapshot-config>` — Roll back all Routing Engines and FPCs to another software version and reboot. Include the `with-old-snapshot-config` option to use the saved configuration that corresponds to the rollback software image.
- `request system software sync (all-versions | current | rollback)` — Synchronize software and configurations from the primary Routing Engine to the other nodes and reboot the other nodes.
- `set system auto-sw-sync enable` — Automatically synchronize the software and the configuration from the primary Routing Engine to a newly added Routing Engine and reboot, when the newly added Routing Engine has a different software version from the rest of the system.

Management Interfaces

On Junos OS Evolved, management interfaces are renamed to accommodate more than one management port per Routing Engine node.

Operational Impact

Management interfaces in Junos OS Evolved do not use the same names as Junos OS (`fxp0`, `em0`, `me0`). Instead, the Junos OS Evolved management interface name format is *device-name.type-port*. For example: `re0:mgmt-0`, `re0:mgmt-1`, `re1:mgmt-0`, `re1:mgmt-1`.

The `show interfaces` output displays the status of all interfaces, including management Ethernet interfaces from both Routing Engines of a dual Routing Engine system.

Routing Engine Firewall Filters

In Junos OS, to control the flow of local packets between the physical interfaces and the Routing Engine, you can apply stateless firewall filters to the input or output of the loopback interface. The loopback interface (lo0) is the interface to the Routing Engine and carries no data packets. In Junos OS, filters applied to the loopback interface apply to both network control traffic and management traffic.

Junos OS Evolved, on the other hand, supports two different filters to control the flow of local packets: one for network control traffic (loopback traffic) and one for management traffic. Thus, filters applied to the loopback interface apply only to network control traffic. You can also apply filters separately to the management interface, which enables you to configure a stricter filter on management traffic.

Operational Impact

In Junos OS Evolved, firewall filters applied to the loopback interface apply only to network control traffic. You must explicitly apply firewall filters to the management interface to filter management traffic. In Junos OS Evolved, management filtering uses Routing Engine filters based on Netfilter, a framework that the Linux kernel provides. As a result, only certain matches and actions are supported. [Table 1 on page 9](#) outlines the Junos OS Evolved filter application.

Table 1: Filter Application for Network Control Traffic and Management Traffic

Interface	Filter Direction	Junos OS Evolved Behavior
lo0	input	Filters are applied at the Packet Forwarding Engine and applied on network ingress traffic.
	output	Filters are applied at the Routing Engine and applied on network egress traffic.
management	input	Filters are applied at the Routing Engine and applied on management ingress traffic.

Table 1: Filter Application for Network Control Traffic and Management Traffic (*Continued*)

Interface	Filter Direction	Junos OS Evolved Behavior
	output	Filters are applied at the Routing Engine and applied on management egress traffic.

Junos OS Evolved Network Stack

Junos OS Evolved runs on native Linux. 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 Junos OS Evolved CLI is designed to overcome these differences. Thus, we recommend that you use CLI commands rather than shell commands for any network operations, particularly for operations that require specifying a routing instance.

If you must perform operations in the Linux shell when using Junos OS Evolved, you need to know about the following routing instances, also known as virtual routing and forwarding instances (VRFs):

- **default**—Handles both WAN and management traffic by default, unless you configure the `mgmt_junos` routing instance.
- **mgmt_junos**—When you configure this routing instance, it puts the management port into its own routing instance, which separates the management traffic from the WAN traffic for the Routing Engine.
- **iri**—Handles control plane traffic (internode communication). In the Junos OS Evolved CLI, this is equivalent to the `__juniper_private1__` routing instance.

Operational Impact

In the Junos OS Evolved shell, you can use the `chvrf` (change VRF) utility to execute a command in the context of a specific routing instance, or VRF. For example:

```
[vrf:none] user@host:~$ chvrf -JU default ping 172.16.1.1
[vrf:none] user@host:~$ chvrf -JU iri ping fpc1
[vrf:none] user@host:~$ chvrf -JU mgmt_junos ping 198.51.100.1
[vrf:none] user@host:~$ chvrf -JU iri ssh re1
```

System Logging

In Junos OS Evolved, each node has the standard `journalctl` tool, which is an interface to retrieve and filter the system journal. System log messages are parsed from the system journal. The `relay-eventd` process runs on all nodes and retrieves events (based on the `syslog` configuration) from the system journal as well as error messages from the different applications and forwards them to the `master-eventd` process. The `master-eventd` process runs on the primary Routing Engine and writes the log messages and errors to disk.

Use the [System Log Explorer](#) application to view or compare system log messages in different releases.

Operational Impact

In Junos OS Evolved there is no `messages` file on the backup Routing Engine. All backup Routing Engine logs are in the `messages` file on the primary Routing Engine node.

System Log Message Format

By default, Junos OS Evolved appends the node name to the hostname in system log messages; Junos OS does not. This action keeps Junos OS Evolved system log messages compliant with RFC5424. However, some monitoring systems may not identify a Junos OS Evolved hostname correctly, because the hostname-node name combination does not match any hostnames in the inventory of hostnames.

Operational Impact

If your monitoring system is not identifying Junos OS Evolved hostnames correctly, you should issue the `set system syslog alternate-format` configuration mode command. This command changes the format of the Junos OS Evolved system log messages. The node name is prepended to the process name in the message rather than appended to the hostname, thereby allowing the monitoring system to identify the hostname correctly.

Tracing Architecture

Junos OS Evolved uses a new tracing architecture. All running applications create trace information, with multiple instances of the same application having their own trace information. The Junos OS Evolved `trace-relay` and `trace-writer` applications coordinate tracing information. The `trace-relay` application runs on local nodes and shares a memory buffer with each application. When a Junos OS Evolved application writes to memory, the `trace-relay` application reads the data directly from memory and sends it to the `trace-writer` applications. A `trace-writer` application runs on each Routing Engine node. It receives the

trace information sent from the trace-relay applications and writes it to the appropriate file in Common Trace Format (CTF).

NOTE: For general monitoring and troubleshooting of devices running Junos OS or Junos OS Evolved, we recommend using standard tools such as CLI `show` commands, system log messages, SNMP, and telemetry data. You should avoid using trace messages for general debugging purposes and long-term solutions because they are subject to change without notice.

Operational Impact

In Junos OS, you enable tracing operations by configuring the `traceoptions` statement at the specific hierarchy level you want to trace. Junos OS Evolved, on the other hand, uses an application-based model, and thus trace messages are logged, viewed, and configured by application. As a result, Junos OS Evolved does not support the `traceoptions` statement at many of the hierarchy levels that Junos OS supports. However, some hierarchy levels, such as those under `[edit protocols]`, still require configuring the `traceoptions` statement to enable trace messages.

Although Junos OS disables global tracing operations for many hierarchy levels by default, some processes log trace messages by default for important events. In contrast, all running applications on Junos OS Evolved create trace information at the `info` level by default.

In Junos OS Evolved, you do not view trace files directly, and you should never add, edit, or remove trace files under the `/var/log/traces` directory because this can corrupt the traces. Instead, you use the `show trace application application-name node node-name` command to read and decode trace messages stored in the trace files.

Relevant CLI Commands

- `show trace application application-name node node-name` — Read and decode trace files.
- `clear trace` — Manually clean up trace files.
- `set system trace application` — Modify trace message configurations at the application level.

How Junos OS Evolved Differs from Junos OS

IN THIS SECTION

- Behavioral Differences Between Junos OS Evolved and Junos OS | 13
- New CLI Statements and Commands (Junos OS Evolved) | 25
- Modified CLI Statements and Commands (Junos OS Evolved) | 33
- Changed CLI Command Output (Junos OS Evolved) | 43
- Removed CLI Statements and Commands (Junos OS Evolved) | 47
- XML Differences Between Junos OS and Junos OS Evolved | 50

In many ways, Junos OS Evolved is the same as Junos OS: Key applications such as the routing, bridging, and management software are 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, the CLI syntax, and CLI and XML output. These differences are indicated throughout the Junos OS documentation. However, this section outlines the differences in one place, for your convenience. If applicable, a link takes you to the place in the Junos OS documentation that covers the item.

For a more detailed overview of the top differences between Junos OS and Junos OS Evolved, see ["Top Differences Between Junos OS Evolved and Junos OS" on page 4](#).

Behavioral Differences Between Junos OS Evolved and Junos OS

Behavioral differences between Junos OS Evolved and Junos OS are ways that the two operating systems act differently in certain circumstances. See [Table 2 on page 13](#).

Table 2: How Junos OS Evolved Behavior Differs from Junos OS

Junos OS Evolved Behavior	Junos OS Behavior	Link to Documentation
Access and Authentication		

Table 2: How Junos OS Evolved Behavior Differs from Junos OS *(Continued)*

Junos OS Evolved Behavior	Junos OS Behavior	Link to Documentation
In Junos OS Evolved Release 20.4R1 and earlier releases, when you do not configure the password authentication method and the remote authentication servers reject the authentication request, the device still attempts local password authentication.	In Junos OS, when you do not configure the password authentication method and the remote authentication servers reject the authentication request, the request ends with the rejection.	<i>Authentication Order for LDAPS, RADIUS, TACACS+, and Local Password</i>
In Junos OS Evolved, you are not able to use remote authentication methods through the console port. You must use the management port for remote authentication.	In Junos OS, you can use remote authentication through the console port and the management port.	<i>Junos OS User Authentication Overview</i>
<p>Junos OS Evolved does not support the following options at the [edit system login retry-options] hierarchy level:</p> <ul style="list-style-type: none"> • backoff-threshold • backoff-factor • maximum-time • minimum-time • tries-before-disconnect 	In Junos OS, the backoff-threshold, backoff-factor, lockout-period, maximum-time, minimum-time, and tries-before-disconnect options are supported at [edit system login retry-options] hierarchy.	<i>retry-options</i>
Interfaces		
The management interface name format changed to accommodate more than one management port per Routing Engine node. The names are re0:mgmt-0/re0:mgmt-1 and re1:mgmt-0/re1:mgmt-1. Both the management interfaces are configurable and displayed.	The management interface name that you use depends on the type of device that you are setting up. Some devices use me0, some use fpx0, and some use em0.	<i>Understanding Management Ethernet Interfaces</i>

Table 2: How Junos OS Evolved Behavior Differs from Junos OS *(Continued)*

Junos OS Evolved Behavior	Junos OS Behavior	Link to Documentation
<p>In an untagged link aggregation group (LAG), child logical interface (IFL) members are created. Requests are made per child IFL member. The results are aggregated and displayed in the CLI.</p> <p>In a VLAN-tagged LAG, extra child IFLs are not created as part of the aggregated Ethernet bundle. Link IFL statistics and marker statistics for child IFLs are not displayed.</p>	<p>Child IFL members are created in untagged and VLAN-tagged LAGs. Requests are made per child IFL member. The results are aggregated and displayed in the CLI.</p>	<p><i>Aggregated Ethernet Interfaces</i></p>
<p>When a new interface is added as a member to an aggregated Ethernet bundle, the new member interface flaps: the physical interface is deleted as a regular interface and then added back in as an aggregated Ethernet member and the statistics are reset.</p>	<p>When a new interface is added as a member to an aggregated Ethernet bundle, that new interface is not first deleted as a lone interface and then added, but everything below it is. Because the interface is not deleted, it keeps all the statistics and other history associated with it.</p>	<p><i>Aggregated Ethernet Interfaces and Understanding Aggregated Ethernet Interfaces and LACP for Switches</i></p>
<p>Junos OS Evolved does not impose a limit on the maximum number of member (or child) interfaces in an aggregated interface. However, platform limits still apply.</p>	<p>Junos OS imposes a limit of 64 member (or child) interfaces in an aggregated interface.</p>	<p><i>Aggregated Ethernet Interfaces and Understanding Aggregated Ethernet Interfaces and LACP for Switches</i></p>
<p>In Junos OS Evolved, when you configure a parent interface for Aggregated Ethernet with the [set interfaces <i>interface-name</i> ether-options 802.3ad <i>ae-name</i>] statement, any secondary (child) interface configurations made from the [edit interfaces <i>interface-name</i>] hierarchy will not take effect until the interface has been committed to the named Aggregated Ethernet (ae) interface. This applies to both ether-options and ggether-options.</p>	<p>In Junos OS, configurations for aggregated Ethernet interfaces and non-aggregated Ethernet interfaces at the [edit interfaces <i>interface</i>] hierarchy are independent of configurations at the [edit interfaces <i>interface</i> ether-options] and [edit interfaces <i>interface</i> ggether-options] hierarchies and will be effective when applied.</p>	<p><i>ether-options, ggether-options</i></p>

Table 2: How Junos OS Evolved Behavior Differs from Junos OS *(Continued)*

Junos OS Evolved Behavior	Junos OS Behavior	Link to Documentation
In Junos OS Evolved, when you add a duplicate IP address or prefix to an existing configuration, the operating system will error out and prevent a commit. Instead, you must first delete the existing prefix and commit the new configuration. After you have done that, you can add the duplicate prefixes and commit.	In Junos OS, you can add a duplicate IP address or prefix to an existing configuration without having your commit blocked.	<i>prefix-list</i>
<p>Starting from Junos OS Evolved Release 21.1R1, we changed the default forward error correction (FEC) for 25-Gigabit and 50-Gigabit interfaces to FEC91 from FEC74 because FEC91 has better performance.</p> <p>FEC mode is assigned by default. You must disable FEC mode if you do not want it assigned by default.</p>	In Junos OS, the default FEC for 25-Gigabit and 50-Gigabit interfaces is FEC74. You can configure FEC clauses CL74 on 25-Gigabit and 50-Gigabit interfaces, and CL91 on 100-Gigabit interfaces. Since the FEC clauses are applied by default on these interfaces, you must disable the FEC clauses if you do not want to apply them.	<i>fec (ether)</i>
High Availability		
On PTX10004 and PTX10008 platforms running Junos OS Evolved, graceful Routing Engine switchover (GRES) is enabled by default and cannot be disabled.	GRES is disabled by default.	<i>Understand Graceful Routing Engine Switchover for Junos OS Evolved</i>
(Only for QFX5220-32CD switches) In-Service Software Upgrade (ISSU) is performed by using the request system software add restart command.	ISSU is performed by using the request system software in-service-upgrade command.	<i>request system software add restart</i>
Junos XML API and Scripting		

Table 2: How Junos OS Evolved Behavior Differs from Junos OS *(Continued)*

Junos OS Evolved Behavior	Junos OS Behavior	Link to Documentation
You must set up the password-less login between two devices to use the <code>jcs:open</code> extension function in SLAX or XSLT scripts to open a connection to the local or remote device.	You are not limited to password-less login. Junos OS supports both a supplied password and interactive password, for example, to execute RPCs on remote devices.	<i>open() Function (SLAX and XSLT)</i>
The <code>eventd</code> process does not give any warning message if there are duplicate event policies. Instead <code>eventd</code> accepts the event policy on a first-come, first-served basis.	The <code>eventd</code> process gives a warning message if you try to create duplicate event policies.	<i>Event Policies and Event Notifications Overview</i>
For <code>op</code> scripts run with the <code>max-datasize</code> configuration statement configured for the minimum memory, an error occurs. In Junos OS Evolved, the error is "Out of memory."	For <code>op</code> scripts run with the <code>max-datasize</code> configuration statement configured for the minimum memory, an error occurs. In Junos OS, the error is "Memory allocation failed."	<i>max-datasize</i>
If you execute the <code>sysctl()</code> extension function in a script and request an invalid <code>sysctl</code> variable name, Junos OS Evolved generates a <code>sysctl</code> error: No such file or directory error.	If you execute the <code>sysctl()</code> extension function in a script and request an invalid <code>sysctl</code> variable name, Junos OS does not generate any error.	<i>Using the sysctl() Extension Function on Junos Devices</i>
Junos OS Evolved stores the trace data for all scripts under the <code>cscript</code> application. The trace log includes data for <code>commit</code> , <code>event</code> , <code>op</code> , and <code>SNMP</code> scripts; <code>YANG</code> action and translation scripts; and <code>Juniper Extension Toolkit</code> scripts. You can modify the default trace settings for all scripts by configuring statements at the <code>[edit system trace application cscript]</code> hierarchy.	Junos OS stores the trace data for each type of script in a different file. You can modify the default trace settings by configuring the <code>traceoptions</code> statement at the hierarchy level for that script type.	<i>Trace Script Processing on Devices Running Junos OS Evolved</i>
Messaging		
TIP: You can compare syslog messages in a Junos OS release to a Junos OS Evolved release using the System Log Explorer.		System Log Explorer

Table 2: How Junos OS Evolved Behavior Differs from Junos OS *(Continued)*

Junos OS Evolved Behavior	Junos OS Behavior	Link to Documentation
The messages file located under /var/log is only written on the primary Routing Engine. Backup Routing Engine messages are found in the messages file on the primary Routing Engine.	The messages file is written on both the primary Routing Engine and the backup Routing Engine.	<i>Displaying System Log Files</i>
Junos OS Evolved appends the node name to the hostname in system log messages. As of Junos OS Evolved Release 20.4R2, you can configure the <code>alternate-format</code> statement at the <code>[edit system syslog]</code> hierarchy level to attach the node name to the process name instead of the hostname. This alternate format allows monitoring systems to identify the hostname correctly.	Junos OS does not.	<i>Overview of System Logging</i>
Starting in Junos OS Evolved Release 20.1R1 and 19.4R2, if you are sending syslog messages to a remote host that is identified by its IP address at the <code>[edit system syslog host ip-address]</code> hierarchy, you only need to configure the <code>management-instance</code> statement to use the <code>mgmt_junos</code> routing instance. You do not need to configure the <code>mgmt_junos</code> routing instance at the <code>[edit system syslog host ip-address routing-instance]</code> hierarchy.	Configure the <code>mgmt_junos</code> routing instance at the <code>[edit system syslog host ip-address routing-instance]</code> hierarchy if you want to send syslog messages to a remote host that is identified by its IP address at the <code>[edit system syslog host ip-address]</code> hierarchy.	<i>routing-instance</i>
When a regular expression returns empty pattern matches, there is no error message.	When a regular expression returns empty pattern matches, you get the following error: <code>regex error: empty (sub)expression</code>	<i>Junos System Log Regular Expression Operators for the match Statement</i>

Routing Policy and Firewall Filters

Table 2: How Junos OS Evolved Behavior Differs from Junos OS *(Continued)*

Junos OS Evolved Behavior	Junos OS Behavior	Link to Documentation
When you issue the <code>show firewall filter ?</code> command, the names of the firewall filters are listed. The names of the Flowspec filters are not listed. To see the names of the configured Flowspec filters, use the <code>show firewall application routing</code> command.	When you issue the <code>show firewall filter ?</code> command, you see not only the names of the firewall filters listed but also the names of the configured Flowspec filters. The Flowspec filters show up inside underscores.	<i>show firewall</i>
Firewall filters applied to the loopback interface apply only to network control traffic. You must explicitly apply firewall filters to the management interface to filter management traffic.	Firewall filters applied to the loopback interface apply to both network control traffic and management traffic.	<i>Stateless Firewall Filter Overview</i>
In Junos OS Evolved, a stateless firewall filter configured on the loopback interface (lo0) does not protect your Routing Engine from TCP and ICMP flood attacks.	In Junos OS, a stateless firewall filter configured on the loopback interface (lo0) protects your Routing Engine from TCP and ICMP flood attacks.	<i>Example: Configuring a Stateless Firewall Filter to Protect Against TCP and ICMP Floods</i>
In Junos OS Evolved, if a match action term on your filter configuration fails on commit, the entire filter is not applied. This happens when a term you configured is not supported on your device.	In Junos OS, if a match action term on your filter configuration fails on commit, the remainder of the filter is applied.	<i>Firewall Filters Overview</i>
When you use an IPv6 filter with packet length matching, the match parameter only considers the TCP header length and the data length. To configure the statement set <code>firewall family inet6 filter filter-name term term-name from packet-length packet-length</code> correctly, you need to specify the packet-length parameter without the IPv6 header size included.	When you use an IPv6 filter with packet length matching, the match length parameter includes the IPv6 header size.	<i>Parameterized Filter Match Conditions for IPv6 Traffic</i>

Table 2: How Junos OS Evolved Behavior Differs from Junos OS (*Continued*)

Junos OS Evolved Behavior	Junos OS Behavior	Link to Documentation
In a filter with icmp match conditions, Junos OS Evolved supports configuration of a single icmp-type value along with an icmp-code value. Junos OS Evolved supports configuration of multiple icmp-type values only when an icmp-code value is not specified.	Junos OS supports a configuration that contains multiple icmp-type values and an icmp-code value.	<i>Overview of Firewall Filters (OCX Series)</i>

Software Installation and Upgrade

Multiple releases of the software can be installed on the device simultaneously as long as there is space. If there is no more space, you must delete an older image of the software before installing the new one.	Only two versions of the software can be installed on the device: the current version and the previous version.	<i>Software Installation and Upgrade Overview (Junos OS Evolved)</i>
The request system snapshot command takes a snapshot of the files currently used to run the device and copies all of these files onto an alternate solid-state drive. The snapshot includes the complete contents of the /soft , /config , and /root directories, copies of user data, and content from the /var directory (except the /var/core , /var/external , /var/log , and /var/tmp directories).	The request system snapshot command takes a snapshot of the contents of the root (/) and /config file systems.	<i>request system snapshot (Junos OS Evolved)</i> <i>Software Installation and Upgrade Overview (Junos OS Evolved)</i>
The request system storage cleanup command does not remove Junos OS Evolved images from the device after Release 20.1R1. It removes all core files, log files from /var/log/ , and all /var/log/* files. To remove old images from the device, use the request system software delete command.	The request system storage cleanup command removes all Junos OS images from the device, including old images and the currently installed image, as well as core files from /var/crash , log files from /var/log/ , and certain other files from /var/tmp .	<i>request system storage cleanup (Junos OS Evolved)</i>

Table 2: How Junos OS Evolved Behavior Differs from Junos OS *(Continued)*

Junos OS Evolved Behavior	Junos OS Behavior	Link to Documentation
Junos OS Evolved supports one system log file that contains all system log messages for the Zero Touch Provisioning (ZTP) process: <code>/var/log/ztp.log</code> .	<p>Junos OS ZTP system log messages are spread out over several system log files:</p> <ul style="list-style-type: none"> • <code>/var/log/dhcp_logfile</code> • <code>/var/log/event-script.log</code> • <code>/var/log/image_load_log</code> • <code>/var/log/messages</code> • <code>/var/log/op-script.log</code> • <code>/var/log/script_output</code> 	<i>Zero Touch Provisioning for Junos OS Evolved</i>
ZTP for Junos OS Evolved supports WAN interfaces as well as the management interface for Routing Engine 0. ZTP dynamically detects the port speed of WAN interfaces and uses this information to create ZTP server ports with the same speed.	ZTP for Junos OS supports management interfaces.	<i>Zero Touch Provisioning for Junos OS Evolved</i>
For ZTP on Junos OS Evolved, if downloading a file fails, ZTP clears the DHCP client binding on that interface and restarts the state machine on other interfaces. If installation fails for any reason, ZTP retries on other interfaces.	If downloading a file fails on Junos OS, the DHCP client attempts to fetch files from the DHCP server for up to six times, with ten to fifteen seconds elapsing between attempts. If the download fails, ZTP stops. ZTP then clears the DHCP client bindings and restarts the state machine on the DHCP-configured interfaces. If installation fails for any reason, ZTP restarts.	<i>Zero Touch Provisioning for Junos OS Evolved</i>
ZTP for Junos OS Evolved accepts unsigned scripts in DHCP option 43, suboption 1.	ZTP for Junos OS with Enhanced Automation accepts unsigned scripts in DHCP option 43, suboption 1; otherwise, scripts must be signed.	<i>Zero Touch Provisioning for Junos OS Evolved</i>

Table 2: How Junos OS Evolved Behavior Differs from Junos OS *(Continued)*

Junos OS Evolved Behavior	Junos OS Behavior	Link to Documentation
ZTP for Junos OS Evolved uses DHCP option 43, suboption 5 for the IP address of the FTP server and does not use option 8.	ZTP for Junos OS uses DHCP option 43, suboption 5 for the HTTP port and uses suboption 8 for the IP address of the HTTP proxy server.	<i>Zero Touch Provisioning for Junos OS Evolved</i>
ZTP for Junos OS Evolved does not change the default route.	For Junos OS, after the lists of bound and unbound client interfaces are created, and a DHCP client gets selected for ZTP activity, any existing default route is deleted and the DHCP client interface that was selected adds a new default route. To add a new default route, only one ZTP instance can be active.	<i>Zero Touch Provisioning for Junos OS Evolved</i>
System Management		
<p>In Junos OS Evolved, the <code>request system reboot</code> command reboots the entire system (all nodes) at once.</p> <p>To reboot a specific node, use the <code>request node reboot</code> command.</p>	In Junos OS, by default, the <code>request system reboot</code> command reboots only the Routing Engine to which you are connected.	<i>request system reboot (Junos OS Evolved)</i>

Table 2: How Junos OS Evolved Behavior Differs from Junos OS (Continued)

Junos OS Evolved Behavior	Junos OS Behavior	Link to Documentation
After rebooting Junos OS Evolved, the system initializes time from the hardware clock. The ntpd command with the -g option runs to adjust the time if the initial offset is large (greater than 1000 seconds). In addition, the system synchronizes time with a valid NTP server.	When you boot Junos OS, the system issues an ntpdate request, which polls a network server to determine the local date and time. You need to configure a server that the system can use to determine the time when the system boots. If an NTP boot server was configured when the system boots, the system immediately synchronizes with the NTP boot server. Synchronization occurs even when the NTP process is explicitly disabled or when the time difference between the client and the NTP boot server exceeds the threshold value of 1000 seconds.	<i>Synchronize and Coordinate Time Distribution Using NTP</i>
Troubleshooting		
Junos OS Evolved uses a new tracing infrastructure. For Junos OS Evolved, trace data from all applications on all nodes is collected on the Routing Engine. You use the show trace application <i>application-name</i> node <i>node-name</i> command to read and decode trace messages stored in the trace files. You can modify trace options for specific applications at the [edit system trace application] hierarchy level. However, a few applications still use the traceoptions statement.	Configure traceoptions to enable trace logging for a specific process or protocol.	<i>trace</i>
For Junos OS Evolved, a core file created during early bootup is stored in /var/core/re . However, a core file created later in the bootup, for example, after the Routing Engine slot number can be determined, is stored in /var/core/re0 or /var/core/re1 . The command show system core-dumps shows all cores generated.	For Junos OS, core files are stored in /var/crash or /var/tmp .	<i>show system core dumps (Junos OS Evolved)</i>

Table 2: How Junos OS Evolved Behavior Differs from Junos OS *(Continued)*

Junos OS Evolved Behavior	Junos OS Behavior	Link to Documentation
User Interface		
Junos OS Evolved does not support the virtual-memory-mapping option.	The virtual-memory-mapping option of the configuration-database statement defines parameters for using virtual memory mapping for the configuration database on a per-process basis.	<i>configuration-database</i>
The show system reboot command has options to Execute this command or Pipe through a command.	The show system reboot command has options to Execute this command, Show halt or reboot requests on both Routing Engines, or Pipe through a command.	<i>show system reboot</i>
In Junos OS Evolved, enabling the command set system switchover-on-routing-crash causes a Routing Engine mastership switchover to occur only on rpd crashes and any uncontrolled rpd exits from outside the CLI (like kill -9 rpi_pid from the Linux shell). Commands like restart routing within the CLI do NOT trigger a switchover.	When NSR is configured and the command edit system switchover-on-routing-crash is enabled, Junos OS will immediately switch to the backup Routing Engine when rpd crashes.	<i>switchover-on-routing-crash</i>
In Junos OS Evolved, when set system processes routing failover other-routing-engine is configured, repeating commands like restart routing and restart routing immediately within the CLI will not cause a Routing Engine mastership switchover when entered more than 4 times in 30 seconds. However, repeated uncontrolled exits (more than 3 times in 5 minutes) from outside the CLI (like rpd crash -9 and rpd kill -15) from the Linux shell will cause rpd to fail and trigger a switchover. If this happens, you must restart the app using the command line interface.	Junos OS triggers a switchover when edit system processes routing failover other-routing-engine is configured and certain commands such as restart routing and restart routing immediately are used many times in short succession.	<i>failover (System Process)</i>

Table 2: How Junos OS Evolved Behavior Differs from Junos OS (Continued)

Junos OS Evolved Behavior	Junos OS Behavior	Link to Documentation
<p>The menu used for root password recovery is the GRUB menu.</p> <p>*Primary ptx-fixed-19.1-16 Primary [Recover password] Primary-Rollback ptx-fixed-19.1-15 Primary-Rollback [Recover password]</p>	<p>The menu used for root password recovery is the Junos Main Menu (the Recovery mode option).</p>	<p><i>Recovering Root Password</i></p>
<p>The <code>show system firmware</code> command displays information based on the accessibility of the device, not the FRU state. The firmware information is cached so, even if the FRU is in a fault condition, the status from the <code>show system firmware</code> command appears as OK. The fault is visible with the commands <code>show chassis alarms</code>, <code>show chassis fpc</code>, and so on.</p>	<p>When the FRU is offline, the cached firmware information of the FRU is not available to view.</p>	<p><i>show system firmware</i></p>

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 more on these new statements and commands, see [Table 3 on page 25](#).

Table 3: New CLI Statements and Commands (Junos OS Evolved)

Statement or Command	Description	Link
New Statements		

Table 3: New CLI Statements and Commands (Junos OS Evolved) *(Continued)*

Statement or Command	Description	Link
<code>[edit system extensions extension-service application file <i>filename</i> interpreter (bash python python3)]</code>	You can use the configuration statement interpreter to specify that a device running Junos OS Evolved run a daemonized on-device JET application using Bash, Python, or Python 3.	<i>file</i>
<code>[edit services monitoring twamp]</code>	<p>You can configure the TWAMP monitoring service on devices running Junos OS Evolved by using the hierarchy level <code>[edit services monitoring twamp]</code>. This service sends out probes to measure network performance. The support for this service is limited to the following:</p> <ul style="list-style-type: none"> • IPv4 and IPv6 traffic (including link-local addresses) for control sessions and test sessions • Control session status and statistics • Test session operational management status and history • Test session probe generation and reception, as well as reflection • Timestamps set by the Routing Engine or the Packet Forwarding Engine • Error reporting through system log messages and SNMP traps only • Unauthenticated mode only 	<i>Understanding Two-Way Active Measurement Protocol on Routers and twamp</i>
<code>[edit security host-vpn]</code>	Junos OS Evolved supports host IPsec in the control plane only (that is, IPsec between the router and external management devices), which is not available in Junos OS. These statements configure a host-to-host VPN type of IPsec connection. Use the <code>connections</code> , <code>ike-log</code> , and <code>ike-secrets</code> statements at the <code>[edit security host-vpn]</code> hierarchy level to configure IKE and IPsec values.	<i>Overview of IPsec and host-vpn</i>

Table 3: New CLI Statements and Commands (Junos OS Evolved) (Continued)

Statement or Command	Description	Link
[edit security host-vpn connections]	<p>You can configure the additional algorithms aes256-sha384-modp3072 and aes256-gcm128-modp3072 at each of the following hierarchy levels:</p> <ul style="list-style-type: none"> [edit security host-vpn connections <i>parent-connection-name</i> ike-proposal] [edit security host-vpn connections <i>parent-connection-name</i> children <i>child-connection-name</i> esp-proposal] 	<i>connections (Host VPN) and children</i>
[edit security host-vpn connections children <i>child-name</i>]	Statements at this hierarchy level include local-traffic-selector, and remote-traffic-selector.	<i>children</i>
[edit security host-vpn connections dpd-delay]	Statement to support dead peer detection. The dead peer detection delay sends keepalives to find out if a peer has gone dead.	<i>connections (Host VPN)</i>
[edit security host-vpn remote]	Configure identity details for authenticating the remote device during IKE negotiations.	<i>remote (Host VPN)</i>
[edit system auto-sw-sync]	Automatically copy over all the images (software and configuration) from the primary Routing Engine of the system to the new Routing Engine and reboot the new Routing Engine so it runs the same software version and configuration as the primary Routing Engine.	<i>auto-sw-sync</i>
[edit system configuration-database extend-size]	<p>Increase the memory space available for the configuration database.</p> <p>NOTE: In some releases prior to Junos OS Evolved Release 22.1R1, the extend-size statement is available in the CLI and you can configure and commit it, but it has no operational effect.</p>	<i>configuration-database</i>

Table 3: New CLI Statements and Commands (Junos OS Evolved) (Continued)

Statement or Command	Description	Link
[edit system log alternate-format]	Attach the node name to the process name instead of the hostname. This alternate format allows monitoring systems to identify the hostname correctly.	<i>syslog</i>
[edit system trace application]	For Junos OS Evolved, trace data from all applications on all nodes is collected on the Routing Engine. See " Top Differences Between Junos OS Evolved and Junos OS " on page 11 for information about tracing architecture. See also the <code>clear trace</code> and <code>show trace</code> commands listed in the New Commands section of this table.	<i>trace</i>
New Commands		
<code>clear node reboot</code>	Remove all pending node halt, reboot, and power-off requests.	<i>clear node reboot</i>
<code>clear security host-vpn security-associations</code>	Clear host IPsec security association information. See also [edit security host-vpn] in the New Statements section of this table.	<i>clear security host-vpn security-associations</i>
<code>clear services monitoring twamp server control-connection</code>	Clear connections established between the Two-Way Active Measurement Protocol (TWAMP) server and control clients.	<i>clear services monitoring twamp server control-connection</i>
<code>clear trace</code>	Junos OS Evolved uses a new tracing infrastructure. This command deletes the trace data stored on the Routing Engine, enabling you to remove inactive tracing sessions. See also [edit system trace application] in the New Statements section of this table.	<i>clear trace</i>

Table 3: New CLI Statements and Commands (Junos OS Evolved) (Continued)

Statement or Command	Description	Link
request node (halt offline online power-off/on reboot) <i>node-name</i>	Request an operation on a specific node.	<i>request node halt (Junos OS Evolved)</i> <i>request node (offline online) (Junos OS Evolved)</i> <i>request node power-off (Junos OS Evolved)</i> <i>request node power-on (Junos OS Evolved)</i> <i>request node reboot (re0 re1) (Junos OS Evolved)</i>
request services monitoring twamp client	Start or stop a Two-Way Active Measurement Protocol (TWAMP) session.	<i>request services monitoring twamp client</i>
request system application restart	Stop and then start (restart) a specific process (for example, cmdd) on the node you specify.	<i>request system application (Junos OS Evolved)</i>
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. Use the node option to collect information from a specific node.	<i>request system debug-info</i>
request system software sync (all-versions current rollback)	Synchronize software and configurations from the primary Routing Engine to the other nodes and reboot the other nodes.	<i>request system software sync</i>

Table 3: New CLI Statements and Commands (Junos OS Evolved) *(Continued)*

Statement or Command	Description	Link
request system software validate restart	The command performs a dry run of the request system software add restart command and displays the ISSU impact of the new restart option. See <i>request system software add (Junos OS Evolved)</i> for more on the restart option.	<i>request system software validate (Junos OS Evolved)</i>
restart <i>app-name</i>	The following message is logged when you use the restart command: App restarting <app name>. Related apps that may be impacted - <related-app name>.	<i>restart (Junos OS Evolved)</i>
show chassis routing-engine hard-disk-test	Display the health of the hard disk with the hard-disk-test option. Use disk /dev/disk-name status argument to display the status of a particular disk.	<i>show chassis routing-engine</i>
show node reboot	Display any pending halt, reboot, or power-off requests on a node.	<i>show node reboot (Junos OS Evolved)</i>
show node statistics	Display the network statistics of a node.	<i>show node statistics</i>
show security host-vpn security-associations	Display host IPsec security association information for a specific security association or for all connections. See also [edit security host-vpn] in the New Statements section of this table.	<i>show security host-vpn security-associations</i>
show security host-vpn version	Display the version of IPsec being used in the system.	<i>show security host-vpn version</i>
show services monitoring rpm history-results	Display the results stored for the specified real-time performance monitoring (RPM) probes.	<i>show services monitoring rpm history-results</i>

Table 3: New CLI Statements and Commands (Junos OS Evolved) *(Continued)*

Statement or Command	Description	Link
<code>show services monitoring rpm probe-results</code>	Display the results of the most recent real-time performance monitoring (RPM) probes.	<i>show services monitoring rpm probe-results</i>
<code>show services monitoring twamp client history-results</code>	Display standard information about the results of the last 50 probes for a Two-Way Active Measurement Protocol (TWAMP) control connection.	<i>show services monitoring twamp client history-results</i>
<code>show services monitoring twamp client probe-results</code>	Display the results of the most recent Two-Way Active Measurement Protocol (TWAMP) probes.	<i>show services monitoring twamp client probe-results</i>
<code>show services monitoring twamp client control-info</code>	Display information about the control connections established between the Two-Way Active Measurement Protocol (TWAMP) server and control clients.	<i>show services monitoring twamp client control-info</i>
<code>show services monitoring twamp client test-info</code>	Display information about the test sessions established between the Two-Way Active Measurement Protocol (TWAMP) server and control clients.	<i>show services monitoring twamp client test-info</i>
<code>show services monitoring twamp server control-info</code>	Display information about the control connections established between the Two-Way Active Measurement Protocol (TWAMP) server and control clients for managed servers.	<i>show services monitoring twamp server control-info</i>
<code>show services monitoring twamp server test-info</code>	Display information about the test sessions established between the Two-Way Active Measurement Protocol (TWAMP) server and control clients.	<i>show services monitoring twamp server test-info</i>
<code>show system applications (app <i>app-name</i> brief detail node <i>node-name</i>)</code>	Display information about active applications on the system.	<i>show system applications (Junos OS Evolved)</i>

Table 3: New CLI Statements and Commands (Junos OS Evolved) *(Continued)*

Statement or Command	Description	Link
show system errors	<p>Display information about faults in the system.</p> <p>NOTE: For Junos OS Evolved, only the QFX5200 supports this command. For all other Junos OS Evolved platforms, use the <i>show system errors active</i>, <i>show system errors count</i>, <i>show system errors error-id</i>, or <i>show system errors fru</i> command.</p>	<i>show system errors</i>
show system errors history	<p>Display information about faults in the system that have been cleared.</p> <p>NOTE: For Junos OS Evolved, only the QFX5200 supports this command. For all other Junos OS Evolved platforms, use the <i>show system errors active</i>, <i>show system errors count</i>, <i>show system errors error-id</i>, or <i>show system errors fru</i> command.</p>	<i>show system errors history</i>
show system nodes	View a list of all nodes in the system.	<i>show system nodes</i>
show system software add-restart	Display all console messages from the last in-service software upgrade (ISSU).	<i>show system software add-restart (Junos OS Evolved)</i>
show system software list	Display the installed versions on all nodes in the system.	<i>show system software list</i>
show system statistics backup	Displays system statistics options for the backup Routing Engine. The options provided are the same as the options for show system statistics.	<i>show system statistics</i>
show system statistics jtd	Displays system jtd statistics.	<i>show system statistics</i>

Table 3: New CLI Statements and Commands (Junos OS Evolved) (Continued)

Statement or Command	Description	Link
<code>show system ztp</code>	Junos OS Evolved implements ZTP using the Linux dhcp client. Users can find out the interfaces chosen by ZTP, arguments returned by DHCP, and ZTP state machine states.	<i>show system ztp</i>
<code>show trace</code>	Junos OS Evolved uses a new tracing infrastructure. This command shows the trace data from all nodes that are collected on the Routing Engine.	<i>show trace</i>
<code>show forwarding-options enhanced-hash-key</code>	Junos OS Evolved uses a new command to display the hashing algorithm to make hashing decisions. This command shows the data about which packet fields are used by the hashing algorithm.	<i>show forwarding-options enhanced-hash-key</i>
<code>show vlans</code>	<p>Junos OS Evolved replaces the <code>show bridge</code> command with the <code>show vlans</code> command. This command displays detailed information on the VLAN configurations present on the Routing Engine and includes the following options:</p> <ul style="list-style-type: none"> • <code>brief</code>: Display brief output. • <code>detail</code>: Display detailed output. • <code>extensive</code>: Display extensive output. • <code>instance</code>: Display information for a specified instance. • <code>interface</code>: Name of interface for which to display table. • <code>logical-system</code>: Name of logical system, or 'all'. • <code>operational</code>: Show operational bridging instance. 	<i>show vlans</i>

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 4 on page 34](#).

NOTE: For the CLI commands that produce changed output, see [Table 5 on page 43](#).

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

Statement or Command	Change in Junos OS Evolved	Link
Modified Statements		
[edit chassis error minor action]	The offline and disable-pfe actions are not available for errors with minor severity.	<i>error</i>
[edit firewall family <i>family-name</i> filter <i>filter-name</i>]	Egress filters do not support gre-key matches.	<i>promote gre-key</i>
[set forwarding options enhanced-hash-key]	The vxlan configuration options are not supported.	<i>show forwarding-options enhanced-hash-key</i>
[edit interfaces <i>interface-name</i> ether-options]	The following options are added to the ether-options statement: <ul style="list-style-type: none"> • fec • loopback-remote 	<i>ether-options</i>
[edit interfaces aggregated-interface-name aggregated-ether-options lacp]	The following options for this command are not supported: <ul style="list-style-type: none"> • accept-data • link-protection • no-peer-loopback-validation 	<i>Configuring Aggregated Ethernet LACP</i>
[edit services monitoring twamp client control-connection test-session offload-type]	In Junos OS Evolved, the option inline-timestamping is configured instead of the hardware-timestamping option as in Junos OS.	<i>test-session (Junos OS Evolved)</i>

Table 4: Modified CLI Statements and Commands (Junos OS Evolved) *(Continued)*

Statement or Command	Change in Junos OS Evolved	Link
[edit system internet-options]	<p>In Junos OS Evolved, the following options are not supported:</p> <ul style="list-style-type: none"> • gre-path-mtu-discovery • icmpv4-rate-limit • icmpv6-rate-limit • ipip-path-mtu-discovery • ipv6-path-mtu-discovery-timeout • no-gre-path-mtu-discovery • no-ipip-path-mtu-discovery • no-ipv6-path-mtu-discovery • no-ipv6-reject-zero-hop-limit • no-source-quench • no-tcp-reset • no-tcp-rfc1323 • no-tcp-rfc1323-paws • source-port • source-quench • tcp-drop-synfin-set 	<i>internet-options</i>
Modified Commands		
clear ipv6 neighbors	In Junos OS Evolved, issuing the clear ipv6 neighbors command clears the cache for IPv6 neighbors <i>that are in a reachable state</i> .	<i>clear ipv6 neighbors</i>

Table 4: Modified CLI Statements and Commands (Junos OS Evolved) *(Continued)*

Statement or Command	Change in Junos OS Evolved	Link
monitor traffic interface	The write-file option for the monitor traffic interface command takes precedence over the extensive option when you configure those two options simultaneously. If you try to configure these options at the same time, Junos OS Evolved gives you a warning message that the options are not compatible, and it only runs the monitor traffic interface write-file command.	<i>monitor traffic</i>
ping	<p>Junos OS Evolved does not support the following ping command options:</p> <ul style="list-style-type: none"> • detail • logical-system • loose-source • mac-address • strict • strict-source • vpls 	<i>ping</i>
request chassis routing-engine master switch	The default wait time on the PTX10008 between Routing Engine switchovers when using the request chassis routing-engine master switch command has increased from 120 seconds to 360 seconds.	<i>request chassis routing-engine master</i>

Table 4: Modified CLI Statements and Commands (Junos OS Evolved) *(Continued)*

Statement or Command	Change in Junos OS Evolved	Link
request system software add	<p>The following request system software add command options are not applicable in Junos OS Evolved:</p> <ul style="list-style-type: none"> • best-effort-load • both-routing-engines • chassis • device-alias • delay-restart • force-host • lcc • member • no-copy • on-primary • (re0 re1) • re-choice • satellite • scc • set • sfc • upgrade-group • unlink • validate • validate-on-host 	<i>request system software add (Junos OS Evolved)</i>

Table 4: Modified CLI Statements and Commands (Junos OS Evolved) *(Continued)*

Statement or Command	Change in Junos OS Evolved	Link
	<ul style="list-style-type: none"> • validate-on-routing-engine 	
request system software delete	<p>The following request system software delete command options are not applicable in Junos OS Evolved:</p> <ul style="list-style-type: none"> • chassis • lcc • member • re-choice • scc • sfc • upgrade-group • unlink • validate • validate-on-host • validate-on-routing-engine 	<i>request system software delete (Junos OS Evolved)</i>

Table 4: Modified CLI Statements and Commands (Junos OS Evolved) *(Continued)*

Statement or Command	Change in Junos OS Evolved	Link
request system software rollback	<p>The following options are added to the request system software rollback command:</p> <ul style="list-style-type: none"> • (no-validate validate) • with-old-snapshot-config <p>The following options are not applicable in Junos OS Evolved:</p> <ul style="list-style-type: none"> • device-alias • satellite • satellite-arg • upgrade-group 	<i>request system software rollback</i>
request system software validate	<p>The following request system software validate command options are not applicable in Junos OS Evolved:</p> <ul style="list-style-type: none"> • chassis • lcc • member • package-options • scc • sfc 	<i>request system software validate (Junos OS Evolved)</i>

Table 4: Modified CLI Statements and Commands (Junos OS Evolved) *(Continued)*

Statement or Command	Change in Junos OS Evolved	Link
request system storage cleanup	<p>Use the new option force-deep to clean up all user-generated files.</p> <p>The user is prompted to check the list of files to be deleted by using the dry-run option.</p> <p>The following options are not applicable in Junos OS Evolved:</p> <ul style="list-style-type: none"> • re0 • re1 • routing-engine 	<i>request system storage cleanup (Junos OS Evolved)</i>
request security pki ca-certificate ca-profile-group load	The default option is not supported on PTX10003-80C, PTX10003-160C, and PTX10008 routers.	<i>request security pki ca-certificate ca-profile-group load</i>
request system zeroize	The local option is removed. The command will reboot all Routing Engines on the local chassis when you issue the command.	<i>request system zeroize</i>
show agent sensors	This command displays output on each Routing Engine, instead of just the primary Routing Engine.	<i>show agent sensors</i>
show chassis fabric summary	<p>More detailed information is provided. The following fields are introduced:</p> <ul style="list-style-type: none"> • Link Error • Link TF • Reachability Errors (Local/Remote) • Uptime 	<i>show chassis fabric summary</i>

Table 4: Modified CLI Statements and Commands (Junos OS Evolved) *(Continued)*

Statement or Command	Change in Junos OS Evolved	Link
show firewall	The application lsp option allows you to specify the display of implicit policers that are published by rpd.	<i>show firewall</i>
show host	The routing-instance mgmt_junos option is introduced.	<i>show host</i>
show system	The nodes and node-attributes options are introduced.	<i>show system nodes, show system node-attributes</i>
show system connections	<p>The node option is introduced.</p> <p>Junos OS Evolved does not support the following show system connections command options:.</p> <ul style="list-style-type: none"> • extensive • show-routing-instance 	<i>show system connections</i>
show system core-dumps	The node option is introduced. The core dump files generated on the nodes are stored in the /var/core/ directory.	<i>show system core-dumps</i>
show chassis errors	The <i>error-id</i> option is moved to the show system errors tree.	<i>show system errors active</i>
show chassis routing-engine errors	The output for this command is moved to show system errors.	<i>show system errors active</i>
show system memory	The node option is introduced.	<i>show system memory</i>

Table 4: Modified CLI Statements and Commands (Junos OS Evolved) *(Continued)*

Statement or Command	Change in Junos OS Evolved	Link
show system processes	<p>The following show system processes command options are not applicable in Junos OS Evolved:</p> <ul style="list-style-type: none"> • health • resource-limits 	<i>show system processes</i>
show system storage	<p>The node option is introduced.</p> <p>The invoke-on option is removed.</p>	<i>show system storage</i>
show system virtual-memory	The node option is introduced.	<i>show system virtual-memory</i>
show version	The node option is introduced.	<i>show version</i>
telnet	<p>Junos OS Evolved does not support the following telnet command options:</p> <ul style="list-style-type: none"> • bypass-routing • interface • logical-system • no-resolve • source 	<i>telnet</i>

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

Statement or Command	Change in Junos OS Evolved	Link
traceroute	<p>Junos OS Evolved does not support the following traceroute command options:</p> <ul style="list-style-type: none"> • logical-system • monitor • next-hop • port • propogate-ttl 	<i>traceroute</i>

Changed CLI Command Output (Junos OS Evolved)

For changes in output for Junos OS Evolved, see [Table 5 on page 43](#).

Table 5: 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.	<i>clear interfaces statistics</i>
monitor traffic interface <i>interface-name</i>	When you use the command monitor traffic interface <i>interface-name</i> on a logical interface, the output displays all packets received or transmitted on that interface, including Layer 2 traffic. When you use this command on a physical interface, the output only displays packets received and transmitted on the physical interface and does not include traffic from the logical interface.	<i>monitor traffic</i>
ping	When pinging a nonresponsive route, the display output of the ping command does not print the number of packets sent or received or the packet loss.	<i>ping</i>

Table 5: Changed Command Output (Junos OS Evolved) (Continued)

Command	Description of Change in Output	Link
request system snapshot	Output displays the names of the directory and the individual files being copied instead of only the directory names.	<i>request system snapshot (Junos OS Evolved)</i>
request system software add	For Junos OS Evolved, this command has a built-in feature to not start an upgrade if a reboot is pending after an upgrade or rollback.	<i>request system software add (Junos OS Evolved)</i>
request system software delete	Output displays the version instead of the package.	<i>request system software delete (Junos OS Evolved)</i>
request system software rollback	Output displays the version instead of the package.	<i>request system software rollback (Junos OS Evolved)</i>
The show chassis environment cb command does not show the Bus and FPGA revision information. Use the show system firmware command in order to view the FPGA revision or version information for the CB.	Use the show chassis environment cb command to display environmental information about the Control Boards (CBs).	<i>show chassis environment cb</i>
show chassis environment fpc	Displays different output.	<i>show chassis environment fpc</i>
show interfaces aenumber extensive	LACP packets and LAG links on the members of an aggregated Ethernet interface are not counted as part of the bundle input or output statistics in the show interfaces aenumber extensive command output.	<i>show interfaces (Aggregated Ethernet)</i>

Table 5: Changed Command Output (Junos OS Evolved) (Continued)

Command	Description of Change in Output	Link
<code>show interfaces</code>	Configuration of IPv6 over the re0:mgmt-* interfaces is supported.	<i>show interfaces</i>
<code>show interfaces detail</code>	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.	<i>show interfaces detail</i>
<code>show interfaces extensive</code>	Output does not display the Packet Forwarding Engine configuration and CoS default bandwidth allocation information.	<i>show interfaces</i>
<code>show interfaces interface-name statistics</code>	Junos OS Evolved does not display statistics for an interface if it is a child of an aggregated ethernet (AE) interface.	<i>show interfaces statistics</i>
<code>show interfaces interface-name ifl-class</code>	Junos OS Evolved does not display statistics for an interface if it is a child of an aggregated ethernet (AE) interface.	<i>show interfaces statistics</i>
<code>show lldp local-information</code>	Output does not display "kernel JUNOS" in the system description field because Junos OS Evolved does not have a kernel.	<i>show lldp local-information</i>
<code>show multicast route extensive</code>	Output displays the Sensor ID field that corresponds to a multicast route.	<i>show multicast route</i>
<code>show multicast usage</code>	Output displays the Sensor ID field that corresponds to a multicast route.	<i>show multicast usage</i>
<code>show policer</code>	Output doesn't display the default ARP policer because it isn't needed in Junos OS Evolved. Distributed denial of service (DDoS) protection replaces the functionality of the default ARP policer.	<i>show policer</i>

Table 5: Changed Command Output (Junos OS Evolved) (Continued)

Command	Description of Change in Output	Link
<code>show snmp mib get</code>	Output for a Routing Engine displays the Routing Engine slot number, not the Routing Engine number.	<i>show snmp mib</i>
<code>show snmp mib walk</code>	The <code>show snmp mib walk jnxFilledDescr</code> output only shows the fan tray number. This output does not show the number of fan slots present in each tray.	<i>show snmp mib</i>
<code>show system errors fru detail</code>	Output displays status of FRUs including CB, chassis, fans, FPC, FPM, PDU, PICS, PSM, RE, and SIB, not just FPC.	<i>show system errors fru</i>
<code>show system memory</code>	Output displays the information per node, and the System memory usage distribution displays only the total, active, inactive, and free memory.	<i>show system memory</i>
<code>show system snapshot</code>	Output displays the snapshot device and a list of snapshots. The list shows the names of the snapshots instead of the version of the operating system. Output does not display the date the snapshot was created.	<i>show system snapshot (Junos OS Evolved)</i>
<code>show system statistics arp</code>	After running ping on an unreachable host, output shows that counts for ARP requests received and for datagrams for an address not on the interface are incremented.	<i>show system statistics arp</i>
<code>show system statistics tcp</code>	Output for the <code>show system statistics tcp</code> command is trimmed to show only fields supported in Junos OS Evolved.	<i>show system statistics tcp</i>
<code>show system uptime</code>	In certain releases, the output displays only the System booted and System-wide users information and does not display information on current time, system booted, protocols started, or last configured parameters. The <code>show system uptime node</code> command shows the other information.	<i>show system uptime</i>
<code>show task replication</code>	Output displays the same state whether the command is run from the primary or the backup Routing Engine.	<i>show task replication</i>

Table 5: Changed Command Output (Junos OS Evolved) (Continued)

Command	Description of Change in Output	Link
show version	<p>Output of the show version command is changed to clearly show which Junos architecture is running on the device.</p> <p>Output of the show version node all command is revised to explicitly identify the Routing Engine in both the XML and CLI output.</p>	<i>show version (Junos OS Evolved)</i>
traceroute	Output of the traceroute command displays MPLS data parsed in the same way as the Linux traceroute command: L=label, E=exp_use, S=stack_bottom, and T=TTL.	<i>traceroute</i>

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 6 on page 47](#). Where there is an alternative statement or command to use, it is noted in the table.

Table 6: Removed CLI Statements and Commands (Junos OS Evolved)

Statement or Command	Description
Removed Statements	
edit forwarding-options analyzer	The analyzer application for port mirroring is not supported on Junos OS Evolved.
edit forwarding-options enhanced-hash-key ecmp-dlb ether-type edit forwarding-options enhanced-hash-key lag-dlb ether-type	On QFX5130 and QFX5700 devices, ether-type is not supported on Junos OS Evolved.
edit system services extension-service notification	Junos OS Evolved does not support the notification service for JET applications.

Table 6: Removed CLI Statements and Commands (Junos OS Evolved) (Continued)

Statement or Command	Description
<code>gigether-options</code>	The <code>gigether-options</code> statement at the <code>[edit interfaces <i>interface-name</i>]</code> hierarchy no longer appears because it is not needed. To configure link aggregation groups (LAG), use the <code>set interfaces <i>interface-name</i> ether-options</code> command instead.
<code>traceoptions</code>	Junos OS Evolved removes or does not support the <code>traceoptions</code> option at many hierarchy levels because trace messages are logged, viewed, and configured per application. However, routing protocols (the <code>[edit protocols]</code> hierarchy level) and a few other applications still use <code>traceoptions</code> .
Removed Commands	
<code>request chassis beacon service-node</code>	This command is removed from Junos OS Evolved.
<code>request system core-dump</code>	This command is removed from Junos OS Evolved.
<code>request system recover</code>	This command is removed from Junos OS Evolved.
<code>request system scripts (delete rollback)</code>	AI-Scripts and Service Now are not supported on Junos OS Evolved.
<code>request system software abort</code>	This command is removed because the <code>request system software add</code> command has a built-in feature not to start an upgrade if a reboot is pending after an upgrade or rollback.
<code>request system software (add delete) set</code>	Junos OS Evolved bundles all packages into one single ISO file, so the <code>set</code> option serves no purpose in the <code>request system software add</code> and <code>request system software delete</code> commands.
<code>request system software in-service-upgrade</code>	Use the <code>request system software add restart</code> command for ISSU. The <code>request system software add</code> command has a built-in feature not to start upgrade if a reboot is pending after an upgrade or rollback.

Table 6: Removed CLI Statements and Commands (Junos OS Evolved) (Continued)

Statement or Command	Description
<code>request system software set</code>	To set the current system to an installed software version, use the <code>request system software rollback reboot</code> command.
<code>request system storage user-disk</code>	There are no satellite packages in Junos OS Evolved.
<code>show bridge</code>	The command <code>show bridge</code> is replaced by the command <code>show vlan</code> in Junos OS Evolved.
<code>show chassis fabric unreachable</code>	See the <code>show system errors</code> command for similar functionality.
<code>show chassis memory-usage-chassisd</code>	The functionality for this command and all options under this command are moved to <code>show system memory</code> .
<code>show chassis network-services</code>	This command is not supported.
<code>show chassis routing-engine errors</code>	This command has been replaced by <code>show system errors</code> in Junos OS Evolved.
<code>show class-of-service forwarding-table</code>	The removed options include <code>classifier</code> , <code>classifier mapping</code> , <code>drop-profile</code> , <code>policer</code> , <code>rewrite-rule</code> , <code>rewrite-rule mapping</code> , <code>scheduler-map</code> , and <code>shaper</code> .
<code>show database-replication</code>	This command is not supported.
<code>show firewall family inet filter filter-name term term-name then traffic-class-count</code>	The <code>traffic-class-count</code> option is not supported under the firewall hierarchy in Junos OS Evolved.
<code>show interfaces mac-database</code>	This command is not supported.
<code>show interfaces mc-ae</code>	This command has been replaced with <code>show multi-chassis mc-lag</code> .

Table 6: Removed CLI Statements and Commands (Junos OS Evolved) (Continued)

Statement or Command	Description
<code>show system buffers</code>	This command is removed starting in Junos OS Evolved Releases 21.1R1 and 20.3R2. This command is not applicable in Junos OS Evolved because the command displays the status of kernel mbufs, which are not used in Linux-based systems like Junos OS Evolved.
<code>show system software detail</code>	Use <code>show system software list</code> to display a list of the software versions installed on all nodes. For more details about the software, use <code>show version detail</code> .
<code>show system uptime invoke-on</code>	This command is removed from Junos OS Evolved.

XML Differences Between Junos OS and Junos OS Evolved

This section lists the differences in XML output between Junos OS and Junos OS Evolved.

request system storage cleanup

In Junos OS, the XML output of `request system storage cleanup` uses the `<file-list>` XML tag for all file types in the list of files to be deleted. In Junos OS Evolved, the XML output groups different file types inside different XML tags, for example, `<core-file-list>` and `<log-file-list>`. Additionally, the command targets all nodes on Junos OS Evolved, so a `<node>` element encloses the output for each node.

request system storage cleanup (Junos OS)

```

user@host> request system storage cleanup | display xml
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/18.4I0/junos">
  <system-storage-cleanup-information>
    <file-list junos:style="normal">
      <file>
        <file-name>/var/log/dfcd_enc.0.gz</file-name>
        <size junos:format="551B">551</size>
        <date>Nov 23 15:33</date>
      </file>
    </file-list>
  </system-storage-cleanup-information>
</rpc-reply>

```

```

    </system-storage-cleanup-information>
</rpc-reply>

```

request system storage cleanup (Junos OS Evolved)

```

user@host> request system storage cleanup | display xml
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/19.1I0/junos">
  <system-storage-cleanup-information>
    <node>
      <node-name> RE0 </node-name>
      <core-file-list>
        <description>List of all core files to be cleared: </description>
        <file>
          <file-name>/var/core/re0/auditd.re.re0.17130.2019_02_28.03_39_36.tar.gz</
file-name>
          <size>3.8M</size>
          <date>Thu Feb 28 03:40</date>
        </file>
      </core-file-list>
      <core-local-host-file-list>
      </core-local-host-file-list>
      <core-subdir-file-list>
      </core-subdir-file-list>
      <fpc-file-list>
      </fpc-file-list>
      <logical-systems-file-list>
      </logical-systems-file-list>
      <log-file-list>
        <description>Clears all App logs, App traces and App SI traces
under /var/log/*, /var/log/traces/* and /var/log/si_traces/* </description>
        </log-file-list>
        <iso-file-list>
        </iso-file-list>
      </node>
    </system-storage-cleanup-information>
  </rpc-reply>

```

show system memory

In Junos OS Evolved, the `show system memory` XML output is changed to better reflect the way Linux manages memory. The output comprises a top-level `<multi-routing-engine-results>` element and one `<multi-`

routing-engine-item> child element for each node, which contains the node name and the <system-memory-information> for that node. In Junos OS, the device only emits a <system-memory-information> element. Additionally, the <system-memory-summary-information> includes the following new child elements:

- <system-memory-used> and <system-memory-used-percent>
- <system-memory-buffer> and <system-memory-buffer-percent>
- <system-memory-swap> and <system-memory-swap-percent>

and omits the following elements:

- <system-memory-reserved> and <system-memory-reserved-percent>
- <system-memory-wired> and <system-memory-wired-percent>
- <system-memory-cache> and <system-memory-cache-percent>

```
user@host> show system memory | display xml
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/22.1R0/junos">
  <multi-routing-engine-results>
    <multi-routing-engine-item>
      <re-name>fpc1</re-name>
      <system-memory-information>
        <system-memory-summary-information>
          <system-memory-total>16125892</system-memory-total>
          <system-memory-total-percent>100%</system-memory-total-percent>
          <system-memory-used>3885112</system-memory-used>
          <system-memory-used-percent>24%</system-memory-used-percent>
          <system-memory-active>2447796</system-memory-active>
          <system-memory-active-percent>15%</system-memory-active-percent>
          <system-memory-inactive>2101128</system-memory-inactive>
          <system-memory-inactive-percent>13%</system-memory-inactive-percent>
          <system-memory-free>9327316</system-memory-free>
          <system-memory-free-percent>57%</system-memory-free-percent>
          <system-memory-buffer>314516</system-memory-buffer>
          <system-memory-buffer-percent>1%</system-memory-buffer-percent>
          <system-memory-swap>2598948</system-memory-swap>
          <system-memory-swap-percent>16%</system-memory-swap-percent>
        </system-memory-summary-information>
        <pmap-terse-information xmlns="http://xml.juniper.net/fbsd10/14.2I0/junos-pmap">
          <pmap-terse-summary junos:style="pmap-process-terse-summary">
            <pid>1</pid>
            <process-name>/lib/systemd/systemd</process-name>
```

```

        <size>159116</size>
        <size-percent>0</size-percent>
        <resident>8408</resident>
        <resident-percent>0</resident-percent>
    </pmap-terse-summary>
    ...
</pmap-terse-information>
</system-memory-information>
</multi-routing-engine-item>
<multi-routing-engine-item>
    <re-name>re0</re-name>
    <system-memory-information>
        <system-memory-summary-information>
            <system-memory-total>16125576</system-memory-total>
            <system-memory-total-percent>100%</system-memory-total-percent>
            <system-memory-used>6912492</system-memory-used>
            <system-memory-used-percent>42%</system-memory-used-percent>
            <system-memory-active>4936580</system-memory-active>
            <system-memory-active-percent>30%</system-memory-active-percent>
            <system-memory-inactive>8939976</system-memory-inactive>
            <system-memory-inactive-percent>55%</system-memory-inactive-percent>
            <system-memory-free>170744</system-memory-free>
            <system-memory-free-percent>1%</system-memory-free-percent>
            <system-memory-buffer>524676</system-memory-buffer>
            <system-memory-buffer-percent>3%</system-memory-buffer-percent>
            <system-memory-swap>8517664</system-memory-swap>
            <system-memory-swap-percent>52%</system-memory-swap-percent>
        </system-memory-summary-information>
        <pmap-terse-information xmlns="http://xml.juniper.net/fbsd10/14.2I0/junos-pmap">
            <pmap-terse-summary junos:style="pmap-process-terse-summary">
                <pid>1</pid>
                <process-name>/sbin/init</process-name>
                <size>162220</size>
                <size-percent>0</size-percent>
                <resident>10780</resident>
                <resident-percent>0</resident-percent>
            </pmap-terse-summary>
            ...
        </pmap-terse-information>
    </system-memory-information>
</multi-routing-engine-item>

```

```
...
</multi-routing-engine-results>
```

show system processes

On certain platforms running Junos OS Evolved Release 20.3R1 or earlier, the XML output for the `show system processes` command and the `show system processes wide` command is the CLI output enclosed in an `<output>` element. Starting in Junos OS Evolved Release 20.4R1, the XML output matches the Junos OS XML output.

show system processes (Junos OS)

```
user@host> show system processes | display xml | no-more
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/20.1R0/junos">
  <system-process-information junos:style="brief">
    <process-information>
      <process>
        <pid>0</pid>
        <terminal-name>- </terminal-name>
        <state>DLs</state>
        <cpu-time>8:39.74</cpu-time>
        <command>[kernel]</command>
      </process>
      <process>
        <pid>1</pid>
        <terminal-name>- </terminal-name>
        <state>ILs</state>
        <cpu-time>0:00.25</cpu-time>
        <command>/sbin/init --</command>
      </process>
      ...
    </process-information>
  </system-process-information>
</cli>
  </banner>
</cli>
</rpc-reply>
```

show system processes (Junos OS Evolved)

```

user@host> show system processes | display xml | no-more
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/20.2I0/junos">
  <output>
    -----
    node: re0
    -----


| UID  | PID | PPID | C | SZ   | RSS  | PSR | STIME   | TTY | TIME     | CMD                    |
|------|-----|------|---|------|------|-----|---------|-----|----------|------------------------|
| root | 1   | 0    | 0 | 9947 | 2732 | 1   | Apr10 ? |     | 00:00:22 | /sbin/init --dump-core |
| root | 2   | 0    | 0 | 0    | 0    | 5   | Apr10 ? |     | 00:00:00 | [kthreadd]             |
| root | 3   | 2    | 0 | 0    | 0    | 0   | Apr10 ? |     | 00:00:20 | [ksoftirqd/0]          |
| root | 5   | 2    | 0 | 0    | 0    | 0   | Apr10 ? |     | 00:00:00 | [kworker/0:0H]         |
| root | 7   | 2    | 0 | 0    | 0    | 5   | Apr10 ? |     | 00:04:20 | [rcu_preempt]          |


    ...
  </output>
</cli>
  </banner>
</cli>
</rpc-reply>

```

show system processes wide (Junos OS)

```

user@host> show system processes wide | display xml | no-more
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/20.1R0/junos">
  <system-process-information junos:style="brief">
    <process-information>
      <process>
        <pid>0</pid>
        <terminal-name>- </terminal-name>
        <state>DLs</state>
        <cpu-time>8:39.86</cpu-time>
        <command>[kernel]</command>
      </process>
      <process>
        <pid>1</pid>
        <terminal-name>- </terminal-name>
        <state>ILs</state>
        <cpu-time>0:00.25</cpu-time>
        <command>/sbin/init --</command>
      </process>

```

```

    ...
  </process-information>
</system-process-information>
<cli>
  </banner>
</cli>
</rpc-reply>

```

show system processes wide (Junos OS Evolved)

```

user@host> show system processes wide | display xml | no-more
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/20.2I0/junos">
  <output>
    -----
    node: re0
    -----
    UID      PID  PPID  C   SZ   RSS  PSR  STIME  TTY      TIME  CMD
    root      1    0    0  9947 2732   0 Apr10 ?      00:00:22 /sbin/init --dump-core
    root      2    0    0    0    0    5 Apr10 ?      00:00:00 [kthreadd]
    root      3    2    0    0    0    0 Apr10 ?      00:00:20 [ksoftirqd/0]
    root      5    2    0    0    0    0 Apr10 ?      00:00:00 [kworker/0:0H]
    root      7    2    0    0    0    0 Apr10 ?      00:04:20 [rcu_preempt]
    ...
  </output>
</cli>
  </banner>
</cli>
</rpc-reply>

```

Default Directories for Junos OS Evolved File Storage

Junos OS Evolved files are stored in the following directories on the device:

- **/boot**—This directory contains the boot loader and associated files.

- **/config**—This directory contains the current operational router or switch configuration and the last three committed configurations, in the files **juniper.conf**, **juniper.conf.1**, **juniper.conf.2**, and **juniper.conf.3**, respectively. The **/config/scripts** directory contains all stored scripts.
- **/data**—This is the directory for all mutable copies of mutable directories. It contains the following subdirectories:
 - **/config**—Contains version-specific Juniper configuration files. This directory is bind mounted to **/config**, meaning that changes in either directory will be reflected in both directories.
 - **/etc**—Contains version-specific Linux configuration files. This directory is bind mounted to **/etc**.
 - **/etc/ssh/ssh**—Contains SSH host keys.
 - **/var**—Shared writable directory for all software versions. This directory is bind mounted to **/var**.
 - **/var_db**—Contains version-specific **/var/db** files. This directory is bind mounted to **/var/db**.
 - **/var_db/scripts**—Contains subdirectories for various script types. Scripts are stored in and executed from these directories. This directory is bind mounted to **/var/db/scripts**.
 - **/var/db/scripts/commit**—Contains commit scripts.
 - **/var/db/scripts/op**—Contains op scripts.
 - **/var/db/scripts/event**—Contains event scripts.
 - **/var/db/scripts/snmp**—Contains SNMP scripts.
 - **/var/db/scripts/lib**—Contains imported scripts.
 - **/var_etc**—Contains version-specific **/var/etc** files. This directory is bind mounted to **/var/etc**.
 - **/var_pfe**—Contains version-specific PFE configuration files. This directory is bind mounted to **/var/pfe**.
 - **/var_rundb**—Contains UI-related runtime-generated database files that are shared across versions. This directory is bind mounted to **/var/rundb**.
- **/soft**—This directory is the software install area. All software versions are installed here.
- **/u**—This directory is a read-only file system for the running version of Junos OS Evolved.
- **/var**—This directory contains the following subdirectories:
 - **/home**—Contains users' home directories, which are created when you create user access accounts. For users using SSH authentication, their **.ssh** file, which contains their SSH key, is placed in their home directory. When a user saves or loads a configuration file, that file is loaded from the current working directory unless the user specifies a full pathname.

- **/db/config**—Contains up to 46 previous versions of committed configurations, which are stored in the files **juniper.conf.4.gz** through **juniper.conf.49.gz**.
- **/log**—Contains system log and tracing files.
- **/core**—Contains core files. The software saves up to five core files, numbered from 0 through 4. File number 0 is the oldest core file and file number 4 is the newest core file. To preserve the oldest core files, the software overwrites the newest core file, number 4, with any subsequent core file.
- **/tmp**—Contains temporary files, including files that are generated when a crash event is detected.

RELATED DOCUMENTATION

| [Junos OS Evolved Overview](#) | 2

Junos OS Evolved Components and Processes

IN THIS SECTION

- [Linux Kernel](#) | 59
- [Initialization Process](#) | 59
- [System Epoch Management Process](#) | 59
- [System Manager Process](#) | 59
- [Management Process](#) | 60
- [Routing Protocol Process](#) | 60
- [Interface Process](#) | 60
- [Distributor Process](#) | 60
- [SNMP and MIB II Processes](#) | 61
- [Process Limits](#) | 61

A Junos OS Evolved system is comprised of one or more Linux nodes, coupled together with an efficient communications substrate, and supplied with a distributed application launcher. A horizontal software layer decouples application processes from the specific hardware node where they can be run.

Applications use the Distributed Data Store (DDS) to share state, and state is synchronized between nodes. A high-level description of the various software components is listed below.

Linux Kernel

Junos OS Evolved is built on top of a stock Linux kernel. Functionality performed by the router like configuration management, interface management and routing are processes that run as Linux processes. All applications run natively on the Linux kernel, including Juniper and non-Juniper applications.

Initialization Process

When the device boots, an initialization process (init) starts and monitors all the other software processes.

If a software process terminates or fails to start when called, the init process attempts to restart it a limited number of times and logs any failure information for further investigation.

System Epoch Management Process

The system epoch management process (SysEpochMan) is responsible for organizing the various Linux nodes into a cohesive system, and to monitor the system to ensure integrity if any nodes fail. If the system needs to be restarted, SysEpochMan ensures a clean transition from the previous system state to the new system state.

System Manager Process

The system manager process (SysMan) is responsible for the launching, coordination, and monitoring of applications in Junos OS Evolved. The SysMan Master oversees the placement of applications on nodes as specified by each application, and communicates its decisions to the local SysMan instances. If an application fails, the local SysMan process will detect the failure, and take corrective action based on what is specific for the application.

Management Process

The management process (mgd) manages the configuration of the router and all user commands. The management process is responsible for managing all user access to the device and for notifying other processes when a new configuration is committed. A dedicated management process handles Junos XML protocol XML requests from its client, which might be the CLI or any Junos XML protocol client.

Routing Protocol Process

Within Junos OS Evolved, the routing protocol process (rpd) controls the routing protocols that run on the device. The rpd process starts all configured routing protocols and handles all routing messages. It maintains one or more routing tables, which consolidate the routing information learned from all routing protocols. From this routing information, the routing protocol process determines the active routes to network destinations and installs these routes into the Routing Engine's forwarding table. Finally, rpd implements routing policy, which enables you to control the routing information that is transferred between the routing protocols and the routing table. Using routing policy, you can filter and limit the transfer of information as well as set properties associated with specific routes.

Interface Process

The Junos OS Evolved interface process (Ifmand) is responsible managing all interfaces on the device. Ifmand creates all the operational state related to interfaces (IFD, IFL, IFF, IFA) as well as the necessary interface specific routes and nexthops.

Ifmand enables you to configure and control the physical interface devices and logical interfaces present in a network device. You can configure interface properties such as the interface location, for example, in which slot the Flexible PIC Concentrator (FPC) is installed and in which location on the FPC the Physical Interface Card (PIC) is installed, as well as the interface encapsulation and interface-specific properties. You can configure the interfaces currently present in the device, as well as interfaces that are not present but that you might add later.

Distributor Process

The distributor process is responsible for holding the Distributed Data Store (DDS) and coordinating with individual applications for delivery of their state. The distributor process synchronizes state across the system.

SNMP and MIB II Processes

Junos OS Evolved supports the Simple Network Management Protocol (SNMP), which helps administrators monitor the state of a device. The software supports SNMP version 1 (SNMPv1), version 2 (SNMPv2, also known as version 2c, or v2c), and version 3 (SNMPv3).

Process Limits

There are limits to the total number of Junos OS Evolved processes that can run simultaneously on a device. There are also limits set for the maximum number of iterations of any single process. The limit for iterations of any single process can only be reached if the limit of overall system processes is not exceeded.

Error TPAs for Route Installation

SUMMARY

If you configure this feature, during route installations the consumer of a state update notifies the producing application when there are errors in processing the state update sent by the producer. The producer then attaches a third-party attachment (TPA) object on top of the errored object, with details of the error, and publishes it.

IN THIS SECTION

- [Overview of Error Third-Party Attachments \(TPAs\) on Errored Objects During Route Installations | 61](#)
- [Set Up the System for Error TPAs | 62](#)
- [CLI Commands for Viewing Error Details | 63](#)

Overview of Error Third-Party Attachments (TPAs) on Errored Objects During Route Installations

In a distributed system, states can be produced anywhere and consumed anywhere, making it difficult for a producer (for example, a PFE) to determine whether the system is in the correct state for the consumer (for example, an rpdagent). If you configure this feature, during route installations the consumer notifies the producing application when there are errors in processing the state update sent

by the producer. The producer then attaches a TPA object on top of the errored object with details of the error and publishes it.

Details of errors include:

- errorID
- severity
- obj_guid
- error_description
- error_module
- error_object_name
- error_timestamp
- error_producer_name
- natural_name

The errors generated have standard error numbers.

The forwarding information base (FIB) telemetry daemon (FIBtd) also receives error notifications. You use the Junos telemetry interface (JTI) and remote procedure calls (gRPC) services to stream or export ON_CHANGE FIB statistics to an outside SDN collector. Set the collector to subscribe to xpath **/state/system/anomalies/fib/** to get both the IPv4 and IPv6 error routes.

You can use the CLI to query errored objects and related information. To avoid flooding the system with error objects, the number of published error objects from a producer is set to a threshold limit of 20,000. Once the threshold is reached, no more error objects are published. However, errored objects and related information is still saved, you can query it using CLI

The consumer is notified when the errors are cleared and the route installation is successful.

Set Up the System for Error TPAs

SUMMARY

- 1. Configure FIP streaming on the client device.

```
set system fib-streaming
set system services extension-service request-response grpc max-connections number
set system services extension-service request-response grpc skip-authentication
set system services extension-service notification allow-clients address ip-address
set system services extension-service request-response grpc clear-text port port-number
```

- 2. On the collector, subscribe to the xpath `/state/system/anomalies/fib/` to get both the IPv4 and IPv6 error routes.

CLI Commands for Viewing Error Details

SUMMARY

Use the following CLI commands to view details of error TPAs that are generated during route installations:

Table 7: CLI Commands to View Error TPA Information

Command	Example	Link
show system applications	show system applications error app rpdagent node re0	<i>show system applications (Junos OS Evolved)</i>
show fib-streaming	show fib-streaming native-model route-errors inet	<i>show fib-streaming</i>
show agent sensors	-	<i>show agent sensors</i>

Shell Commands for Junos OS Evolved

IN THIS SECTION

- [How to Use the Shell | 64](#)
- [Common Shell Commands | 64](#)

Shell commands are Linux commands that are executed through the Linux shell rather than the Junos OS Evolved CLI. Junos OS Evolved supports existing Linux shell commands. This topic lists commonly used shell commands for Junos OS Evolved.

How to Use the Shell

To start the Linux shell, enter the `start shell` command from the Junos OS Evolved CLI. When you are in the shell, the command prompt will change to the following format:

```
username@hostname: ~$
```

Once the shell is active, you can enter shell commands using the shell prompt. To return to the Junos OS Evolved CLI, use the `exit` command.

Common Shell Commands

The following table lists some of the shell commands that are useful for operating a Junos OS Evolved device:

Table 8: Junos OS Evolved Shell Commands

Command	Description
sync	Synchronize the Routing Engines This command should only be used in situations where the CLI cannot be accessed.
reboot	Reboot the current Routing Engine. This command should only be used in situations where the CLI cannot be accessed.
/sbin/upgrade /var/tmp/iso	Upgrade the current Routing Engine using the specified .iso file. This command should only be used in situations where the CLI cannot be accessed.
vssh <i>node-name</i>	Open a SSH session to the remote node from the Routing and Control Board.
chvrf iri <i>network-command</i>	Creates the network context required to access the control plane and reach other nodes.
systemctl enable --now docker.service	Enable automatic startup for the Docker container service
who	Displays a list of users logged into the device. Hostname is displayed for users connected via telnet and IP address is displayed for users connected via SSH.
ecmp-tracer -- <i>interface</i> &	Displays the forwarding packets going through an interface.

RELATED DOCUMENTATION

[Junos OS Evolved Overview](#) | 2

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.

- [Getting Started with Junos OS Evolved](#)—Procedures for initial configuration.
- [User Access and Authentication Administration Guide for Junos OS Evolved](#)—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.
- [Junos® OS Evolved Software Installation and Upgrade Guide](#)—Procedures for installing and upgrading Junos OS Evolved software.
- [CLI User Guide for Junos OS Evolved](#)—Procedures on using the CLI for Junos OS Evolved software.

2

CHAPTER

Junos OS Evolved Configuration Overview

Junos OS Evolved Configuration Basics | 68

Methods for Configuring Junos OS Evolved | 68

Junos OS Evolved Configuration from External Devices | 71

Junos OS Evolved Configuration Basics

Your compatible Juniper Networks device comes with Junos OS Evolved installed on it, unless you specifically order it without the operating system. When Junos OS Evolved is pre-installed, you simply power on the device and all software starts automatically. You just need to configure the device so it will be ready to participate in the network.

To configure the Junos OS Evolved, you must specify a hierarchy of configuration statements which define the preferred software properties. You can configure all properties of the Junos OS Evolved, including interfaces, general routing information, routing protocols, and user access, as well as some system hardware properties. After you have created a candidate configuration, you commit the configuration to be evaluated and activated by Junos OS Evolved.

RELATED DOCUMENTATION

[Junos OS Evolved Configuration from External Devices | 71](#)

[Methods for Configuring Junos OS Evolved | 68](#)

[Junos OS Evolved Overview | 2](#)

Methods for Configuring Junos OS Evolved

IN THIS SECTION

- [Junos OS Evolved Command-Line Interface | 69](#)
- [ASCII File | 70](#)
- [Junos XML Management Protocol Software | 70](#)
- [NETCONF XML Management Protocol Software | 70](#)
- [Configuration Commit Scripts | 70](#)

Depending on specific device support, you can use the methods shown here to configure Junos OS Evolved. For more information, see the [Juniper Networks Feature Explorer](#).

Table 9: Methods for Configuring Junos OS Evolved

Method	Description
Command-line interface (CLI)	Create the configuration for the device using the CLI. You can enter commands from a single command line, and scroll through recently executed commands.
ASCII file	Load an ASCII file containing a configuration that you created earlier, either on this system or on another system. You can then activate and run the configuration file, or you can edit it using the CLI and then activate it.
Junos XML management protocol (API)	Client applications use the Junos XML management protocol to monitor and configure Juniper Networks devices. The Junos XML management protocol is customized for Junos OS Evolved, and operations in the API are equivalent to those in the Junos OS Evolved CLI.
NETCONF application programming interface (API)	Client applications use the NETCONF XML management protocol to monitor and configure supported devices. The NETCONF XML management protocol includes features that accommodate the configuration data models of multiple vendors.
Configuration commit scripts	Create scripts that run at commit time to enforce custom configuration rules. Commit scripts are written in Python, Stylesheet Language Alternative syntaX (SLAX), or Extensible Stylesheet Language Transformations (XSLT).

The following sections describe the methods you can use to configure Junos OS Evolved:

Junos OS Evolved Command-Line Interface

The Junos OS Evolved CLI is a straightforward terminal-based command interface. You use Emacs-style keyboard sequences to move around on a command line and scroll through a buffer that contains recently executed commands. You type commands on a single line, and the commands are executed when you press the Enter key. The CLI also provides command help and command completion.

ASCII File

You can load an ASCII file containing a configuration that you created earlier, either on this system or another system. You can then activate and run the configuration file as is, or you can edit it using the CLI and then activate it.

Junos XML Management Protocol Software

The Junos XML Management Protocol is an XML-based protocol that client applications use to monitor and configure Juniper Networks devices. It uses an XML-based data encoding for the configuration data and remote procedure calls. This API is customized for Junos OS Evolved, and operations in the API are equivalent to CLI commands.

NETCONF XML Management Protocol Software

The NETCONF XML management protocol is an XML-based protocol that client applications use to monitor and configure network devices. It uses an XML-based data encoding for the configuration data and remote procedure calls. NETCONF includes features that accommodate the configuration data models of multiple vendors. Juniper Networks provides a set of Perl modules that enable Perl client applications to communicate with the NETCONF server on Junos devices. The Perl modules enable you to develop custom applications for configuring and monitoring Junos OS Evolved.

Configuration Commit Scripts

You can create and use scripts that run at commit time to enforce custom configuration rules. If a configuration breaks the custom rules, the script can generate actions that the Junos OS Evolved performs. These actions include:

- Generating custom error messages
- Generating custom warning messages
- Generating custom system log messages
- Making changes to the configuration

Configuration commit scripts also enable you to create macros, which expand simplified custom aliases for frequently used configuration statements into standard Junos OS Evolved configuration statements. Commit scripts are written in Python, Stylesheet Language Alternative syntaX (SLAX), or Extensible Stylesheet Language Transformations (XSLT).

RELATED DOCUMENTATION

[CLI Explorer](#)

[CLI User Guide](#)

[Junos OS Evolved Configuration from External Devices | 71](#)

[NETCONF XML Management Protocol Developer Guide](#)

[Junos OS Evolved Overview | 2](#)

Junos OS Evolved Configuration from External Devices

You can configure a Junos OS Evolved network device from a *system console* connected to the console port or by using *Telnet* to access the device remotely. External management hardware can be connected to the Routing Engine and the Junos OS Evolved through these ports:

- Console port
- Auxiliary port
- Ethernet management port

NOTE: See hardware guide for your particular Junos OS Evolved device for instructions about how to connect external hardware to the console, auxiliary, and/or Ethernet management ports. Capabilities and features can vary depending on device model.

RELATED DOCUMENTATION

[Methods for Configuring Junos OS Evolved | 68](#)

[Junos OS Evolved Overview | 2](#)

3

CHAPTER

Running 3rd Party Applications with Junos OS Evolved

Overview of Third-Party Applications on Junos OS Evolved | 73

Managing Third-Party Applications | 77

Running Third-Party Applications in Containers | 88

Signing Third-Party Applications to Run Natively on Junos OS Evolved | 93

Building Third-Party Applications | 99

Overview of Third-Party Applications on Junos OS Evolved

IN THIS SECTION

- [Introduction to Third-Party Applications on Junos OS Evolved | 73](#)
- [Running Applications in Containers vs Using Signing Keys | 73](#)
- [Application Pre-requisites | 74](#)
- [Application APIs | 75](#)
- [Security Caveats | 76](#)
- [File Security with IMA | 76](#)

Introduction to Third-Party Applications on Junos OS Evolved

Junos OS Evolved runs natively on Linux, which means you can integrate third-party applications and tools developed for Linux into Junos OS Evolved. Linux development tools also give you the power to create and run your own applications on Junos OS Evolved. You can choose to run these applications inside a container, or natively on the device with signing keys.

Running Applications in Containers vs Using Signing Keys

IN THIS SECTION

- [Applications in Containers | 74](#)
- [Using Signing Keys | 74](#)

There are two ways to run a third-party applications in Junos OS Evolved: running inside a container, or running natively using signing keys.

Applications in Containers

Junos OS Evolved supports running applications inside Docker containers. Containers run on Junos OS Evolved, and applications run inside the containers, keeping them isolated from the OS. You can use prebuilt Docker container images and install additional tools and libraries inside the container. Containers can be upgraded by using Linux workflow.

Containers are already a commonly used method for running Linux applications, so many existing third-party applications can be easily imported into Junos OS Evolved by deploying them inside containers. The isolated nature of containers makes them easy to deploy and remove without compromising the integrity of Junos OS Evolved. In addition, Junos OS Evolved places default limits on the resource usage of containers, to ensure that rogue containers cannot overwhelm your system.

The Docker container service is not automatically started at system initialization. To enable automatic startup for the Docker container service, enter the following command from the Linux shell:

- `# systemctl enable --now docker.service`

For more information about running applications in containers, see ["Running Third-Party Applications in Containers" on page 88](#)

Using Signing Keys

The other method of running third-party applications on Junos OS Evolved is by using signing keys. You can generate signing keys and use them to sign executable files or shared objects. Signing an executable file gives it permission to run on the device, allowing you to approve trusted applications to run alongside authorized Juniper Networks software.

Signing keys are controlled by a Linux subsystem called Integrity Measurement Architecture (IMA). IMA policy consists of rules that define which actions need to be taken before a file can be executed. IMA measurement policy will measure and store a file's hash, and IMA appraisal policy will make sure that the file has a valid hash or digital signature. IMA will only allow a file to run if this validation succeeds.

Junos OS Evolved requires users to sign all files that will be mapped into memory for execution. IMA verification helps ensure that these files have not been accidentally or maliciously altered. Containers and files inside containers do not need to be signed.

For more information about using signing keys, see ["Signing Third-Party Applications to Run Natively on Junos OS Evolved" on page 93](#)

Application Pre-requisites

Third party application support was introduced with Junos OS Evolved release 20.1R1, so in order to install and use third party applications you must be running Junos OS Evolved release 20.1R1 or later.

Applications must support the Linux kernel version running on Junos OS Evolved to work properly. Use the `show version` command to view the currently running Linux kernel version.

Applications written for Junos OS Evolved typically require the ability to read and modify the networking state, to send and receive packets, and to read and modify the configuration. Junos OS Evolved supports a limited number of APIs, so applications must be configured with these APIs in mind.

Application APIs

There are two categories of APIs used by applications:

- Linux APIs for reading and modifying the networking state, and sending and receiving packets.
- Juniper APIs for interacting with the system.

Junos OS Evolved supports these two categories of APIs. [Table 10 on page 75](#) provides a high-level view of the set of APIs used by applications:

Table 10: Application APIs

<i>API</i>	<i>Functionality</i>
Packet IO and Linux socket APIs	Ability to send and receive packets over mgmt and/or data interfaces. Standard libc – send, receive, listen, etc.
rtnetlink	Ability to use rtnetlink to query networking state like interfaces, routes, etc.
netdevice	Ability to configure network devices.
proc	Ability to query kernel data structures using standard interfaces provided by Linux kernel.
Junos APIs	Ability to access Juniper Northbound APIs - NetConf/JET/Telemetry.

NOTE: For more information on Juniper Northbound APIs, see the following:

- [Overview of JET APIs](#)
- [NETCONF XML Management Protocol and Junos XML API Overview](#)
- [Overview of the Junos Telemetry Interface](#)

Security Caveats

Junos OS Evolved is designed from the ground up with security in mind. IMA and Linux containers help to control the security impact of third-party applications on Junos OS Evolved, but third-party applications still have the potential to introduce security vulnerabilities through malicious code.

Always consider the security implications of adding a third-party application to Junos OS Evolved. Make sure any applications you add to Junos OS Evolved are thoroughly vetted for potential security risks.

File Security with IMA

Network devices that run Junos OS Evolved are protected by an integrity solution called Integrity Measurement Architecture (IMA).

Integrity is a fundamental security property that represents trust, completeness, and freedom from alteration. In computer security, common targets for integrity protections are operating system files. A common method of ensuring integrity is to compare a file against a known good file.

In the context of Junos OS Evolved, the security goal is to ensure that the software running on a device has not been accidentally or maliciously altered. The software running on a device is either authentic Junos software from Juniper Networks or authorized software deployed by a customer.

The threat model for network devices includes attempts by malicious actors to deploy malware that violates either the implicit or explicit policies of device owners. Such malware could include back doors, Trojan horses, or implants that could adversely affect the safe and secure operation of devices or networks. Malicious actors use a variety of tools, techniques, and procedures to breach integrity including physical attacks, local attacks, and remote attacks.

Many regulatory schemes levy file integrity requirements, including PCI-DSS - Payment Card Industry Data Security Standard (Requirement 11.5), SOX - Sarbanes-Oxley Act (Section 404), NERC CIP - NERC CIP Standard (CIP-010-2), FISMA - Federal Information Security Management Act (NIST SP800-53 Rev3), HIPAA - Health Insurance Portability and Accountability Act of 1996 (NIST Publication 800-66) and the SANS Critical Security Controls (Control 3).

In order to ensure file integrity and to mitigate the malware risk, Junos OS Evolved runs IMA, and a companion mechanism: the Extended Verification Module (EVM). These open source protections are part of a set of Linux Security Modules that are industry-standard and consistent with the trust mechanisms specified by the Trusted Computing Group.

Juniper Networks applies digital signatures to Junos OS Evolved files, and allows customers to apply digital signatures as well. Digital signatures are created using protected private keys, and then verified using public keys embedded into one or more keyrings.

The IMA/EVM subsystem protects the system by performing run-time checks. If a file fails verification, it is not opened or executed.

That means that unverified software is blocked on a device running Junos OS Evolved.

Managing Third-Party Applications

SUMMARY

IN THIS SECTION

- [How to Run Applications | 77](#)
- [Using Intercept Libraries | 78](#)
- [Removing Third-Party Applications | 87](#)

How to Run Applications

Applications can be launched by using Linux syntax for execution:

```
user@host:~# ./ima-test  
Hello, World!
```

Using Intercept Libraries

SUMMARY

IN THIS SECTION

- [Example of a Preloaded Linux Command | 79](#)
- [Interface Name Translation | 84](#)
- [Caveats for the Intercept Feature | 86](#)

Junos OS Evolved can run third-party applications because it runs on native Linux. 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 uses an intercept mechanism that redirects shell requests for network topology information to a space where the information can be obtained from Junos OS. This intercept mechanism is accomplished through intercept libraries, `libsi.so` and `libnli.so`, that you preload. After you preload the intercept library, certain types of requests are intercepted and show Junos OS information.

The intercept libraries are optional; they are needed only if the application requires the APIs mentioned in [Table 11 on page 78](#):

Table 11: APIs That Require Intercept Libraries

API	Description
Packet IO and Linux socket APIs	Ability to send and receive packets over management and/or data interfaces. Standard libc, such as send, receive, listen.
rtnetlink	Ability to use rtnetlink to query networking state like interfaces, routes.
netdevice	Ability to configure network devices.
proc	Ability to query kernel data structures using standard interfaces provided by Linux kernel.

Table 11: APIs That Require Intercept Libraries (Continued)

API	Description
Junos APIs	Ability to access Juniper North Bound APIs - NetConf/JET/Telemetry.

NOTE: For more information on Juniper Northbound APIs, see the following:

- [Overview of JET APIs](#)
- [NETCONF XML Management Protocol and Junos XML API Overview](#)
- [Overview of the Junos Telemetry Interface](#)

NOTE: Junos OS Evolved Release 20.1R1 supports the following features:

- Use the `set system netlink-async-mode` configuration to enable NETLINK_ROUTE asynchronous notifications. This feature is disabled by default. Use `show nsld mode` to show the current netlink asynchronous mode.
- `SIOCETHOOL ioctl`, which can be used by other applications.
- Multipath next-hop route information through netlink route attributes.

Example of a Preloaded Linux Command

An example how 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 12 on page 85](#)) 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 12: 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 13 on page 85](#) for examples of Junos logical interface names and their Linux-compliant forms.

Table 13: 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

Table 13: Examples of Translated Logical Interface Names (Continued)

Junos Logical Interface Name	Translated Linux-Compliant Interface Name
et-1/2/255:6.7	et-012ff060007
et-1/2/4:5.32767	et-01204057fff
re0:mgmt-1.2	mgmt-0-01-0002
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)
```

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.

Removing Third-Party Applications

There are several methods for removing third-party applications. The method you should use is based on how you installed the application.

- If a third-party application was installed with the `request system software add` command, then you can remove the same application by using the `request system software delete` command.

```
user@host> request system software delete ima-test
Removing version 'ima-test'.
Software ... done.
Data ... done.
Version 'ima-test' removed successfully.
```

- If a third-party application was installed by copying binaries, then you need to know the location of the installed binaries and the key used to sign them.

The first step in removing these applications is to unlink the key with the `request security system-keystore unlink key` command.

```
user@host> request security system-keystore unlink key
```

Next, remove any binaries that you installed for the application with the `rm -f /path/to/binary1 /path/to/binary2` shell command.

```
user@host:~# rm -f /path/to/binary1 /path/to/binary2
```

- If a third-party application was installed through a Docker container, then use the following Docker command to remove the container:

```
docker rm container-name
```

Running Third-Party Applications in Containers

IN THIS SECTION

- Deploying a Docker Container | 89
- Managing a Docker Container | 90
- Enabling Netlink or Packet IO in a Container | 90
- Selecting a VRF for a Docker Container | 91
- Modifying Resource Limits for Containers | 92

To run your own applications on Junos OS Evolved, you have the option to deploy them inside a Docker container. The container runs on Junos OS Evolved, and the agents run inside the container, keeping them isolated from the OS. Containers are installed in a separate partition mounted at `/var/extensions`.

NOTE: Docker containers are not integrated into Junos OS Evolved, they are created and managed entirely through Linux by using Docker commands. For more information on Docker containers and commands, see the official Docker documentation: <https://docs.docker.com/get-started/>

Containers have default limits for the resources that they can use from the system:

- **Storage** – The size of the `/var/extensions` partition is platform driven: 8GB or 30% of the total size of `/var`, whichever is smaller.
- **Memory** – Containers have a default limit of 2GB or 10% of total physical memory, whichever is smaller.
- **CPU** – Containers have a default limit of 20% max CPU use across all cores.

NOTE: You can modify the resource limits on containers if necessary. See "[Modifying Resource Limits for Containers](#)" on page 92.

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. Set the following setenv variable:

```
[vrf:vrf0] user@host_RE0:~# export DOCKER_HOST=unix:///run/docker-vrf0.sock
```

3. Import the image.

NOTE: The URL for the import command needs to be changed for different containers.

```
[vrf:vrf0] user@host_RE0:~# docker import http://198.0.2.2/lxc-images/images/pyez_new/2.1.9/
amd64/default/20190225_19:53/rootfs.tar.xz
```

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

```
[vrf:vrf0] user@host_RE0:~# docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
<none>	<none>	738c70533604	59 seconds ago	491MB

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

```
[vrf:vrf0] user@host_RE0:~# docker create -it --name pyez1 --network=host 738c70533604 bash
```

NOTE: Docker containers are daemonized by default unless you use the -it argument.

Managing a Docker Container

Docker containers are managed through Linux workflow. Use the `ps` or `top` Linux commands to show which Docker containers are running, and use Docker commands to manage the containers. For more information on Docker commands, see: <https://docs.docker.com/engine/reference/commandline/cli/>

NOTE: Junos OS Evolved high availability features are not supported for custom applications in Docker containers. If an application has high availability functionality then you should run the application on each RE to ensure it can sync itself.

Enabling Netlink or Packet IO in a Container

You need to provide additional arguments to Docker commands if your container requires extra capabilities like Netlink or Packet IO. The following example shows how to activate Netlink or Packet IO capabilities for a container by adding arguments to a Docker command:

1. Create a read-only name persistent volume upon starting Docker services:

```
--mount source=jnet,destination=/usr/evo
```

2. Share the host's network namespace with the container process:

```
--network=host
```

3. Automatically start the container upon system reboot:

```
--restart=always
```

4. Enable net admin capability, which is required by Netlink and Packet IO libraries:

```
--cap-add=NET_ADMIN
```

5. Enable the environmental variables required for Netlink and Packet IO:

```
--env-file=/run/docker/jnet.env
```

Selecting a VRF for a Docker Container

Containers inherit virtual routing and forwarding (VRF) from the Docker daemon. In order to run containers in a distinct VRF, a Docker daemon instance needs to be started in the corresponding VRF. The `docker@vrf.service` instance allows for starting a daemon in the corresponding VRF. If the VRF is unspecified, the VRF defaults to `vrf0`.

The `docker.service` runs in `vrf:none` by default.

The docker daemon for a specific VRF listens on corresponding socket located at `/run/docker-vrf.sock`.

The Docker client gets associated with the VRF specific docker daemon by use the following arguments:

```
--env-file /run/docker-vrf/jnet.env
--host unix:///run/docker-vrf.sock or export DOCKER_HOST=unix:///run/docker-vrf.sock
```

For example, to run a container in `vrf0` enter the following Docker command and arguments:

```
[vrf:none] user@host:~#docker -H unix:///run/docker-vrf0.sock run --rm -it --network=host --cap-add=NET_ADMIN --mount source=jnet,destination=/usr/evo --env-file=/run/docker-vrf0/jnet.env
debian:stretch ip link
1002: et-010000000000: BROADCAST,MULTICAST,UP mtu 1514 state UP qlen 1
    link/ether ac:a:a:18:01:ff brd ff:ff:ff:ff:ff:ff
1001: mgmt-0-00-0000: BROADCAST,MULTICAST,UP mtu 1500 state UP qlen 1
    link/ether 50:60:a:e:08:bd brd ff:ff:ff:ff:ff:ff
1000: lo0_0: LOOPBACK,UP mtu 65536 state UP qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
```

NOTE: A container can only be associated to a single VRF.

Modifying Resource Limits for Containers

The default resource limits for containers are controlled through a file located at `/etc/extensions/platform_attributes`. You will see the following text upon opening this file:

```
## Edit to change upper cap of total resource limits for all containers.
## applies only to containers and does not apply to container runtimes.
## memory.memsw.limit_in_bytes = EXTENSIONS_MEMORY_MAX_MIB + EXTENSIONS_MEMORY_SWAP_MAX_MIB:-0
## check current defaults, after starting extensions-cglimits.service
## $ /usr/libexec/extensions/extensions-cglimits get
## please start extensions-cglimits.service to apply changes here

## device size limit will be ignored once extensionsfs device is created
#EXTENSIONS_FS_DEVICE_SIZE_MIB=
#EXTENSIONS_CPU_QUOTA_PERCENTAGE=
#EXTENSIONS_MEMORY_MAX_MIB=
#EXTENSIONS_MEMORY_SWAP_MAX_MIB=
```

To change the resource limits for containers, add values to the EXTENSIONS entries at the bottom of the file:

- `EXTENSIONS_FS_DEVICE_SIZE_MIB=` controls the maximum storage space that containers can use. Enter the value in bytes. The default value is 8GB or 30% of the total size of `/var`, whichever is smaller.
- `EXTENSIONS_CPU_QUOTA_PERCENTAGE=` controls the maximum CPU usage that containers can use. Enter a value as a percentage of CPU usage. The default value is 20% max CPU use across all cores
- `EXTENSIONS_MEMORY_MAX_MIB=` controls the maximum amount of physical memory that containers can use. Enter the value in bytes. The default value is 2GB or 10% of total physical memory, whichever is smaller.



CAUTION: Before modifying the resource limits for containers, be aware of the CPU and memory requirements for the scale you have to support in your configuration. Exercise caution when increasing resource limits for containers to prevent them from causing a strain on your system.

Signing Third-Party Applications to Run Natively on Junos OS Evolved

IN THIS SECTION

- [Signing Keys Overview | 93](#)
- [Generating Signing Keys | 94](#)
- [Importing Signing Keys into the System Keystore and IMA Extended Keyring | 96](#)
- [Viewing the System Keystore and IMA Extended Keyring | 97](#)
- [How to Sign Applications | 98](#)
- [How to Run Signed Applications | 99](#)

Signing Keys Overview

Starting in Junos OS Evolved Release 20.1R1, you can generate signing keys and use them to sign executable files or shared objects. Signing an executable file gives it permission to run on the device, allowing you to approve trusted applications to run alongside authorized Juniper Networks software.

Junos OS Evolved requires users to sign all files that will be mapped into memory for execution. This includes the following file types:

- Executable and Linkable Format (ELF) files
- Shared Objects (.so) files

The following types of files do not need to be signed:

- Docker containers
- Applications inside containers
- Scripts

NOTE: Although scripts don't need to be signed, they do need to be passed through a signed interpreter for execution. Junos OS Evolved comes installed with signed Python 2 and Python 3 interpreters that can be used through the `python script-name` shell command.

Signing keys are controlled by a Linux subsystem called Integrity Measurement Architecture (IMA). IMA policy consists of rules that define which actions need to be taken before a file can be executed. IMA measurement policy will measure and store a file's hash, and IMA appraisal policy will make sure that the file has a valid hash or digital signature. IMA will only allow a file to run if this validation succeeds. For more information about IMA, see ["Protecting the Integrity of Junos OS Evolved with IMA" on page 76](#).

Signing keys are stored in the *system keystore*, and the certificates used to verify signing keys are stored in the *IMA extended keyring*. Keep reading to learn how to generate, import, view, and use signing keys.

Generating Signing Keys

IN THIS SECTION

- [Generating Signing Keys Using the OpenSSL Command-Line | 94](#)
- [Generating Signing Keys Using an OpenSSL Configuration File | 95](#)

Keys can be generated through the OpenSSL command-line or a OpenSSL configuration file.

Generating Signing Keys Using the OpenSSL Command-Line

The following example OpenSSL command can be used to generate signing keys:

```
openssl req -new \
  -newkey rsa:3072 \      # Create an RSA 3072 key
  -x509 \                 # Need an X509 certificates
  -sha256 \               # Strong hashing algorithm
  -nodes \                # No encrypted private-key
  -out ima-cert.x509 \    # Name of the certificate file
  -outform DER \          # Key in DER format
  -keyout privkey.pem \   # Name of the private key
```

This command will generate 2 files:

1. `privkey.pem` - The PEM encoded private key that can be used to sign executable files.
2. `ima-cert.x509` - The DER encoded certificate to be loaded into the IMA extended keyring.

NOTE: The OpenSSL command-line is limited in its functionality. It does not allow you to set values for the X509v3 extensions. All keys generated using the command above can be used as Certificate Authorities (CAs), and therefore can be used to sign other certificates. To prevent this, we can use an OpenSSL Configuration File.

Generating Signing Keys Using an OpenSSL Configuration File

Create a file named `ima-x509.cnf` and paste the following contents:

```
# Beginning of ima-x509.cnf
[ req ]
default_bits = 2048
distinguished_name = custom_distinguished_name
prompt = no
string_mask = utf8only
x509_extensions = custom_exts

[ custom_distinguished_name ]
O = Juniper Networks, Inc.
CN = IMA extended signing key
emailAddress = john.smith@juniper.net

[ custom_exts ]
basicConstraints=critical,CA:FALSE
keyUsage=digitalSignature
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid
# EOF
```

After the configuration file is created, use the following OpenSSL command to create the `ima-privkey.pem` and `ima-cert.x509` files:

```
openssl req -new \
    -nodes \
    -utf8 \
```

```
-sha1 \
-days 36500 \
-batch \
-x509 \
-config ima-x509.cnf \
-outform DER -out ima-cert.x509 \
-keyout ima-privkey.pem
```

The private key file `ima-privkey.pem` is used to generate signing keys, and the certificate file `ima-cert.x509` is used to verify the signature. Both files are used during the process of importing signing keys into the system keystore and IMA extended keyring.

Importing Signing Keys into the System Keystore and IMA Extended Keyring

Signing keys need to be imported into the system keystore prior to use. Keys that are imported into the system keystore are automatically imported into the IMA extended keyring.

To import a signing key into the system keystore, use the `request security system-keystore import` command with the following 3 mandatory arguments:

1. `key-name` - A unique name for the key
2. `private-key` - Path to the private key file
3. `x509-cert` - Path to the DER encoded certificate file

The following example command will create a key named **ima-test-key** by using the private key file `ima-privkey.pem` and the certificate file `ima-cert.x509`:

```
user@host> request security system-keystore import key-name ima-test-key private-key ima-privkey.pem x509-cert ima-cert.x509
```

```
Key Name:          ima-test-key
Private Key Path:   /etc/ima-ext/ima-test-key/privkey.pem
X509 Cert Path:    /etc/ima-ext/ima-test-key/ima-cert.x509
Key SKI:           b71b35e380517cd224b46072dadeb6c53e0a58a1
```

When the key is successfully imported into the system-keystore you will see the above output displaying the name of the key, the paths to the private key and certificate on disk, and the Subject Key Identifier

(SKI) for the key. You can check if this SKI matches with the key loaded into the IMA Extended keyring with the following command:

```
user@host> show security integrity extended-keyring

Keyring
 351716837 ---lswrv      0      0 keyring: ima_ext
 684930381 --als--v     0      0 \_ asymmetric: Juniper Extended Signing Key:
 b71b35e380517cd224b46072dadeb6c53e0a58a1
```

Viewing the System Keystore and IMA Extended Keyring

You can view the contents of the system keystore and the IMA extended keyring through Junos OS Evolved CLI `show` commands.

Use the `show security integrity system-keystore` command to view the available signing keys in the system keystore:

```
user@host> show security integrity system-keystore

Available signing keys:
---
Key Name:          ima-test-key
Private Key Path:  /etc/ima-ext/ima-test-key/privkey.pem
X509 Cert Path:   /etc/ima-ext/ima-test-key/ima-cert.x509
Key SKI:          b71b35e380517cd224b46072dadeb6c53e0a58a1
---
Key Name:          test-key1
Private Key Path:  /etc/ima-ext/test-key1/privkey.pem
X509 Cert Path:   /etc/ima-ext/test-key1/ima-cert.x509
Key SKI:          332f173d61bba03fed5399a609523cbd3cfe66b3
---
Key Name:          test-key2
Private Key Path:  /etc/ima-ext/test-key2/privkey.pem
X509 Cert Path:   /etc/ima-ext/test-key2/ima-cert.x509
Key SKI:          26ebafd58b54f7b8b530d0311503fd84873ee754
---
```

The information in the Key SKI field can be used to map these keys to the IMA extended keyring.

Use the `show security integrity extended-keyring` command to view the contents of the IMA extended keyring:

```
user@host> show security integrity extended-keyring

Keyring
 351716837 ---lswrv      0      0 keyring: ima_ext
 684930381 --als--v     0      0 \_ asymmetric: Juniper Extended Signing Key:
b71b35e380517cd224b46072dadeb6c53e0a58a1
 316767440 --als--v     0      0 \_ asymmetric: Juniper Extended Signing Key:
26ebafd58b54f7b8b530d0311503fd84873ee754
 950431262 --als--v     0      0 \_ asymmetric: Juniper Extended Signing Key:
332f173d61bba03fed5399a609523cbd3cfe66b3
```

How to Sign Applications

After a signing key has been imported into the system keystore, it can be used to sign executable binaries.

Use the `request security integrity measure file filename key key-name` command to sign a file.

The following example command shows a file named **ima-test** being signed by a key named **ima-test-key**:

```
user@host> request security integrity measure file ima-test key ima-test-key
Successfully signed file /data/var/home/root/ima-test
```

You can verify that your file was successfully signed by using the `request security integrity appraise file filename key key-name` command, as follows:

```
user@host> request security integrity appraise file ima-test key ima-test-key
File /data/var/home/root/ima-test has a valid IMA signature
```

If the file was not signed properly, the following message will display:

```
user@host> request security integrity appraise file ima-test key ima-test-key
warning: IMA signature verification failed for /data/var/home/root/ima-test using ima-test-
```

```
key
IMA appraisal for /data/var/home/root/ima-test failed.
```

After a file has been signed, it can be run natively on your Junos OS Evolved device.

How to Run Signed Applications

On attempting to execute a file that has not been signed, you may get a Permission Denied error:

```
user@host:~# ./ima-test
-sh: ./ima-test: Permission denied
```

Once the file has been successfully signed, it can then be executed from a shell prompt by adding the `./` prefix in front of the filename:

```
user@host:~# ./ima-test
Hello, World!
```

Building Third-Party Applications

SUMMARY

IN THIS SECTION

- [JET SDK for Junos OS Evolved | 100](#)
- [SysMan and systemd Controlled Applications | 101](#)
- [Folder Structure for Third-Party Applications | 102](#)
- [Third-Party Application Files | 103](#)
- [Creating a Third-Party Package | 107](#)

Junos OS Evolved supports packages of signed third-party applications developed in C, C++, and Python with the JET SDK for Junos OS Evolved. These third-party applications can be controlled by the Junos

OS Evolved application manager (SysMan) or the Linux system manager (systemd) based on the nature and usage of the application.

JET SDK for Junos OS Evolved

IN THIS SECTION

- [Downloading the JET SDK | 100](#)
- [Installing the JET SDK for Junos OS Evolved | 101](#)

Before building a third-party application to run on your device, you must first generate signing keys to give your application permission to run. See ["Generating Signing Keys" on page 94](#) for more information.

Junos OS Evolved release 22.4R1 and later versions support the JET SDK for Junos OS Evolved on the following platforms:

Table 14: JET SDK for Junos OS Evolved Supported Platforms

ACX Series	PTX Series	QFX Series
ACX7100-32C	PTX10001-36MR	QFX5130-32CD
ACX7100-48L	PTX10003	QFX5130-48C
ACX7509	PTX10004	QFX5220
	PTX10008	
	PTX10016	

Downloading the JET SDK

Before you begin building a third-party application, download and install the JET SDK for Junos OS Evolved and the JET Toolkit for Junos OS Evolved. You can find these files on the [Juniper Networks Downloads](#) page for your device:

- The JET SDK for Junos OS Evolved: **jet-junos-evo-sdk-22.4R1.10-EVO.deb**

- The JET Toolkit for Junos OS Evolved: [jet-junos-evo-toolkit-22.4R1.10-EVO.tar.gz](#)

Installing the JET SDK for Junos OS Evolved

After downloading the JET Toolkit for Junos OS Evolved, unpack the tar file and extract the Jet-evo utility from the package.

After downloading the SDK, install and source it. The JET SDK for Junos OS Evolved is a Debian package that can be installed on Debian-based Linux distributions like Ubuntu and Debian. Use the following Linux command to install the JET SDK for Junos OS Evolved release 22.4R1:

```
user@host:~$ dpkg -i jet-junos-evo-sdk-22.4R1.10-EVO.deb
```

After you run the command, the SDK will be installed into the `/opt/jnpr/sdk/3.0.2-31` folder.

Next, you need to source the SDK for your third-party package. Use the following commands to source the JET SDK for Junos OS Evolved for a package in Junos OS Evolved release 22.4R1:

```
user@host:~/package-name$ source /opt/jnpr/sdk/3.0.2-31/environment-setup-core2-32-poky-linux
user@host:~/package-name$ source /opt/jnpr/sdk/3.0.2-31/environment-setup-core2-64-poky-linux
user@host:~/package-name$ source /opt/jnpr/sdk/3.0.2-31/environment-setup-x86-pokymllib32-linux
```

SysMan and systemd Controlled Applications

Third-party applications can be controlled by the native Junos OS Evolved application manager (SysMan), or the Linux system manager (systemd). SysMan controlled applications can take advantage of Junos OS Evolved's distributed infrastructure and run across multiple nodes. Meanwhile, systemd managed applications can only run on a single node, so SysMan is the preferred application manager in most cases.

SysMan and systemd managed applications use different methods to define policy through service files. Applications managed through SysMan use `.yaml` files to define policy and assign services. See ["yaml Files" on page 105](#) for more information.

Applications managed through systemd define policy and assign services through `.service` files. See ["Service Files" on page 104](#) for more information.

Folder Structure for Third-Party Applications

IN THIS SECTION

- [root-fs Folder Structure | 102](#)
- [Post-install Folder Structure | 102](#)

Third-party application packages follow a specific folder structure in Junos OS Evolved.

root-fs Folder Structure

The following folders are contained under the root-fs folder:

- **usr/sbin** (mandatory): Contains the binary executable file for the application.
- **usr/lib64** (optional): Contains the necessary libraries that are used by the application.
- **etc/systemd/system** (optional): Contains systemd service unit files for systemd controlled applications (mandatory for systemd managed applications).
- **usr/conf** (optional): Contains .yaml files for policy generation (mandatory for SysMan managed applications)

The following optional folders are located outside the root-fs folder:

- **scripts** (optional): Contains the preinstall, post-install and pre-uninstall scripts.
- **etc/config** (optional): Contains application specific configurations.

Post-install Folder Structure

After the third-party package is installed, the files will be stored at the following paths by default:

- **/usr/sbin/**: Contains the binary executable files for applications in the package.
- **/usr/lib64/**: Contains the libraries that are used by applications in the package.
- **/data/var/external/current-evo-version/**: Contains pre-install, post-install, and pre-uninstall script files, as well as .bom, .sh, .fs and .ima files for the applications in the package.
- **/etc/systemd/system/**: Contains .service files for setting application policies and attached services.

- **/usr/conf/**: Contains .yaml files that are used by SysMan controlled applications for setting policies and attached services.

NOTE: By default, the files for libraries and binaries are located in read-only folders. We do not recommend placing libraries and binaries in read-write enabled folders like **/etc** or **/var**, as this makes them vulnerable to being overwritten or modified.

Third-Party Application Files

IN THIS SECTION

- [Makefile | 103](#)
- [SRC Files | 103](#)
- [Script Files | 103](#)
- [Service Files | 104](#)
- [.yaml Files | 105](#)

Third-party application packages are composed of several types of files. Keep reading for more information about the different file types:

Makefile

The Makefile is a necessary file for building and installing your third-party application. The Makefile needs to include the paths for other files like the SRC file, configuration file, and .yaml file. It also controls where the necessary files will be installed.

SRC Files

SRC files contain the source code for your application. SRC files can be written in C, C++ or Python. These files are necessary for the function of your third-party application.

Script Files

Script files include pre-install, post-install and pre-uninstall scripts. These optional files are shell scripts that you can customize to fit the needs of your application.

- Pre-install scripts are run before installing your third-party package. These scripts are typically used for testing and checking various factors on your device. For example you could create a pre-install script to check the version of Junos OS Evolved that is running before starting installation.
- Post-install scripts are run after your installing your third-party package. These scripts contain that commands to run after finishing installation. For example, creating additional log files or a logging directory for your package.
- Pre-uninstall scripts are run before uninstalling your third-party package. These scripts are typically used to clean up folders and files created during the installation of your third-party package.

NOTE: All scripts must return zero upon successful execution and non-zero upon failure.

Scripts can be created in any folder. During the third-party package creation process with the Jet-evo tool, you can enter the paths to each type of script and they will be installed in the `/data/var/external/current-evo-version/` folder on your device. For more information on the Jet-evo tool, see ["Jet-evo Tool Configuration" on page 108](#).

NOTE: If you are creating custom scripts, we recommend that you do not use any bash commands that will create additional CLI output. You can redirect CLI output to your own log file.

Service Files

Service files are mandatory files for systemd controlled applications. These files control the Linux services that are attached to a third-party application.

Service files must be placed in the `etc/systemd/system/` folder inside the `root-fs` directory for your package.

The following example is a `.service` file for a service named `service1`. This `.service` file defines the path to start the service, and the system state required to run the service.

```
[Unit]
Description="Service1 Daemon"

[Service]
ExecStart=/usr/sbin/service1

[Install]
WantedBy=multi-user.target
```

.yaml Files

.yaml files are mandatory files for SysMan controlled applications. These files are used to pass application specific policies that are used in the creation of systemd based service units and SysMan based policies during installation.

Third-party packages can have more than one service attached with the package. Each application controlled by SysMan must have a separate .yaml file for each attached service, with the service names as the .yaml filenames. .yaml filenames should match with the service names, rather the package name, because third-party package names can be different from the services included with the package.

.yaml files must be placed in the **usr/conf** folder inside the **root-fs** directory for your package.

A .yaml file consists of multiple fields containing information about the corresponding application. See the following list for definitions of the .yaml file fields:

binpath	Defines the path to the binary executable file.
exec-start	Defines how the application is started, including the path and the arguments required to start application. This field corresponds to the ExecStart field in a systemd service unit.
exec-stop	Defines the path to the application that runs when the main application is stopping, and the arguments that should be applied. This field corresponds to the ExecStop field in a systemd service unit.
working-dir	Defines the path to the parent directory where binaries are stored.
id	Controls the name of the application. Should match the .yaml filename.
network	Sets the network type, with values of <code>internal</code> or <code>external</code> . Applications that communicate outside the device should use the <code>external</code> value.
on-exit	Controls the setting for restarting the application upon exit. <ul style="list-style-type: none"> • <code>restart</code>: Controls whether the application will be restarted upon exit or not with <code>true</code> or <code>false</code> values.
resource	Sets the limits to system resources used by the application: <ul style="list-style-type: none"> • <code>max-memory</code>: Controls the maximum amount of memory allowed to be consumed by each instance of application. • <code>node-attribute</code>: Controls the type of node that the application will run on, with values like <code>-re</code> and <code>fpc</code>.

- *node-type*: Control the type of node for the action to run on, with values such as `-re` and `fpc`.
- *startup*: Controls if the application will be run by default on bootup or not with `true` or `false` values.
- *instances*: Controls the nodes that the application will be running on and maximum number of instances it can run.
 - *all_nodes*: Control if the application will run on all nodes with `true` or `false` values.
 - *max_num_of_instances*: Control the max number of instances that an application can run with a number value.

The following sample `.yaml` file shows the configuration for a sample application that will run on the primary Routing Engine with only 1 instance. This can be achieved by marking the `all_nodes` field as `false` and `max_number_of_instances` as 1 with `node-attribute` set to `-re`. With this configuration, if the primary Routing Engine goes down, the application will automatically start on the backup Routing Engine.

```
description: "Sample app"
binpath: /usr/sbin/sample/sample_app.py
exec-start: "/usr/bin/python /usr/sbin/sample/sample_app.py"
exec-stop: "/usr/sbin/exit_foo.py --bar"
working-dir: /usr/sbin
id: sample-app
network: internal
on-exit:
  restart: true
resource:
  instances:
    all_nodes: false
    max_num_of_instances: 1
  node-attributes: - re
  max-memory: 2G
  startup: true
```

Creating a Third-Party Package

IN THIS SECTION

- [Create a SysMan Managed Package | 107](#)
- [Create a systemd Managed Package | 107](#)
- [Jet-evo Tool Configuration | 108](#)
- [Installing a Third-Party Package | 109](#)

Third-party packages are created by a tool called Jet-evo using the applications, binaries, and libraries created by the application developer. The process for creating a third-party package differs based on whether the package will be managed by the Junos OS Evolved SysMan process or the Linux systemd process. See below for more details on each specific scenario:

Create a SysMan Managed Package

External applications that are managed by SysMan need to have .yaml files for the services attached to the application.

1. Build and compile the application with C, C++, or Python using the JET SDK for Junos OS Evolved.
2. Create the necessary folders as explained in ["Folder Structure for Third-Party Applications" on page 102](#).
3. Keep the binary executable in the **usr/sbin/** folder.
4. Keep the .yaml files in the **usr/conf/** folder and configuration files in the **etc/config/** folder.
5. Run the Jet-evo tool using options specific to your package. See ["Jet-evo Tool Configuration" on page 108](#) for more information about the configuration options in the Jet-evo tool.

Create a systemd Managed Package

External applications that are managed by systemd need to have a .service file for the services attached to the application.

1. Build and compile the application with C, C++, or Python using the JET SDK for Junos OS Evolved.
2. Create the necessary folders as explained in ["Folder Structure for Third-Party Applications" on page 102](#).
3. Keep the binary executable in the **usr/sbin/** folder.
4. Keep the service file in the **etc/systemd/system/** folder.

5. Run the Jet-evo tool using options specific to your package. See ["Jet-evo Tool Configuration" on page 108](#) for more information about the configuration options in the Jet-evo tool.

Jet-evo Tool Configuration

IN THIS SECTION

- [Required Arguments | 108](#)
- [Optional Arguments | 108](#)
- [Example Syntax | 109](#)

The Jet-evo tool is used to name your third-party package, set the version number, run installation scripts, and set the installation location for your third-party package. These variables are all controlled by options set when you run the jet-evo command:

```
$ ./Jet-evo -h
usage: Jet-evo [-h] -n NAME -r ROOT [-v VERSION] [-a ARCH] [-N NODE]
              [-i PREINSTALL] [-p POSTINSTALL] [-u PREUNINSTALL]
              [-d DIRECTORY] [-x] [-t] [-l LOGFILE] [-k KEY]
```

NOTE: See ["Generating Signing Keys" on page 94](#) for information on generating a private key to use with the Jet-evo tool.

Required Arguments

The following arguments are required:

- n *name*** Set the name of your third-party package.
- r *root*:** Define the root paths to the applications included in your package, separated by commas.
- v *version*:** Set a version number for your package.

Optional Arguments

The following arguments are optional:

-h	Displays a help message explaining the Jet-evo fields.
-a <i>architecture</i>	Choose the architecture targeted by your third-party package.
-n <i>node</i>	Define the nodes targeted by your third-party package.
-i <i>pre-install</i>	Specify file paths to pre-install scripts.
-p <i>post-install</i>	Specify file paths to post-install scripts.
-u <i>pre-uninstall</i>	Specify file paths to pre-uninstall scripts.
-d <i>directory</i>	Set the target directory for installing your package.
-x	Strips all debug files from the package.
-t	Create a TAR file with all the output of the Jet-evo tool.
-l <i>name</i>	Set the name and path of the packager logging file.
-k <i>key</i>	Private key used for signing the third-party package.

Example Syntax

The following is an example of the syntax for the jet-evo command using a package named `sample_package` with version 1.0.1:

```
$ jet-evo -n sample_package -v 1.0.1 -t
-i ./src/scripts/sample_package_preinstall
-p ./src/scripts/sample_package_postinstall
-u ./src/scripts/sample_package_preuninstall
-d ./target/ -r ./install/
-k ima-privkey.pem
```

Installing a Third-Party Package

After you have built or obtained a third-party application package, you can install it onto a Junos OS Evolved device. A package can contain multiple applications that will all be installed together. If an application in your package is already installed on the device, then the existing application will be overwritten by your newly installed package. We recommend keeping applications inside packages unique across different packages.

Before installing a third-party application, you must first install the appropriate signing keys and certificates on all REs. Installation will fail if signing keys are not properly generated. See ["Generating Signing Keys" on page 94](#) for more information.

Use the `show security integrity extended-keyring` command to confirm that keys have been installed properly.

Third-party software can be installed using `request system software add package-name` command. For example:

```
request system software add sample_package.1.0.1.tgz
```

When a third-party package is successfully installed, all associated files will be copied into `/data/var/external/current-evo-version/` folder. The `show version` output will display all of the third-party packages installed for a particular software version under the `External Software` field:

```
user@host> show version
Hostname: sample_host
Model: ptx10008
Junos: 22.4I20221214083306-EVO
Yocto: 3.0.2
Linux Kernel: 5.2.60-yocto-standard-gae998d995
JUNOS-EVO OS 64-bit [junos-evo-install-ptx-x86-64-22.4I20221214083306-EVO]
External Software:
JET app sample_package 1.0.1
```

You can also use the `show software list` command to display all the versions of Junos OS Evolved that are installed on your device, and the third-party packages installed for the current version on each node:

```
user@host> show system software list | no-more
-----
node: re0
-----
Active boot device is primary: /dev/vda
List of installed version(s) :

  '-' running version
  '>' next boot version after upgrade/downgrade
  '<' rollback boot version
  '*' deleted JSU version

> junos-evo-install-ptx-x86-64-22.4I20221214085042 - [2022-12-14 09:14:39]
- junos-evo-install-ptx-x86-64-22.4I20221214083306 - [2022-12-14 08:58:24]
< junos-evo-install-ptx-x86-64-22.4I20221214072149 - [2022-12-14 08:04:20]
```

```
junos-evo-install-ptx-x86-64-22.4I20221214055215 - [2022-12-14 06:39:46]
```

External Software:

JET app sample_package 1.0.1

node: rel

Active boot device is primary: /dev/vda

List of installed version(s) :

'-' running version

'*' deleted JSU version

```
> junos-evo-install-ptx-x86-64-22.4I20221214085042 - [2022-12-14 09:16:48]
```

```
- junos-evo-install-ptx-x86-64-22.4I20221214083306 - [2022-12-14 08:58:59]
```

```
< junos-evo-install-ptx-x86-64-22.4I20221214072149 - [2022-12-14 08:04:52]
```

```
junos-evo-install-ptx-x86-64-22.4I20221214055215 - [2022-12-14 06:40:38]
```

External Software:

JET app sample_package 1.0.1

4

CHAPTER

Finding Software Documentation for Junos OS Evolved

[Where to Find Software Documentation for Junos OS Evolved | 114](#)

[Authentication and Access Control | 115](#)

[Chassis Features | 116](#)

[Class of Service | 118](#)

[Ethernet Switching and Bridging | 120](#)

[Forwarding Options | 123](#)

[High Availability | 124](#)

[Interfaces | 126](#)

[IPv6 | 129](#)

[Multicast | 131](#)

[Multiprotocol Label Switching Applications | 133](#)

[Network Management and Monitoring | 135](#)

[Network Security | 138](#)

[Routing Options | 140](#)

[Routing Policies and Firewall Filters | 142](#)

[Routing Protocols | 145](#)

[Services Applications | 148](#)

[System Basics | 150](#)

Where to Find Software Documentation for Junos OS Evolved

SUMMARY

Learn where to find software documentation for Junos OS Evolved in the Juniper Networks TechLibrary.

Looking for Junos OS Evolved documentation? You've come to the correct place! Learn where to find software documentation for Junos OS Evolved in the Junos OS documentation set.

In general, Juniper features and technologies work the same on Junos OS and Junos OS Evolved, so much of the documentation applies to both operating systems. Given the incredible number of features and amount of documentation available, we want to help you and other Junos OS Evolved users find the most relevant content quickly.



In a few cases, Junos OS and Junos OS Evolved differ. Where possible, we inserted inline notes into the documentation like this to highlight the difference or differences.

In this guide, we've collected recommended documentation topics that apply to Junos OS and Junos OS Evolved. If you look to the left on your screen, you'll see a list of chapters. We organized chapters by Feature Families such as Authentication and Access Control, Routing Protocols, Virtual Private Networks (VPNs), and so on. (Note that in the PDF version, you can display the same view by expanding the table of contents.) Each chapter contains links to related content to help you get the documentation you need.

Alternatively, you can access these same lists from the [Junos OS Evolved Software Documentation](#) page.

In addition, the following resources specific to Junos OS Evolved will help you get up and running quickly:

- Feature Explorer—Use [Feature Explorer](#) to view and compare the software features supported on Junos OS Evolved according to your software release and platform.
- Release Notes—Check out the [Release Notes](#) page to obtain the Release Notes for your version of Junos OS Evolved. Learn about new features, known issues, and more!

- Software Guides—Use these OS-specific guides to help you learn about the basics of Junos OS Evolved:
 - [CLI User Guide for Junos OS Evolved](#)
 - [Day One+ for Junos OS Evolved \(Quick Start\)](#)
 - [Getting Started with Junos OS Evolved](#)
 - [Interfaces Fundamentals for Junos OS Evolved](#)
 - [Introducing Junos OS Evolved](#)
 - [Junos OS Evolved Software Installation and Upgrade Guide](#)
 - [User Access and Authentication Administration Guide for Junos OS Evolved](#)

Authentication and Access Control

SUMMARY

Learn where to find authentication and user access documentation for Junos OS Evolved.

IN THIS SECTION

- [General | 116](#)
- [Related Guide | 116](#)

In general, Juniper features and technologies work the same on Junos OS and Junos OS Evolved, so much of the documentation applies to both operating systems. Given the incredible number of features and amount of documentation available, we want to help you find the most relevant content quickly.

Below we've collected recommended documentation topics and guides that apply to Junos OS and Junos OS Evolved. Alternatively, you can access these same lists from the [Junos OS Evolved Software Documentation](#) page.

NOTE: The documentation provided in this chapter doesn't apply to all Junos OS Evolved platforms and all Junos OS Evolved software releases. For a more detailed list of Junos OS Evolved platforms and the individual features each supports, see [Feature Explorer](#).

General

- [IPv6 Stateless Address Auto-configuration \(SLAAC\) Snooping](#)
- [Mapping OpenConfig AAA Commands to Junos Operation](#)
- [password \(Login\)](#)
- [RADIUS Authentication](#)
- [TACACS+ Authentication](#)

Related Guide

[User Access and Authentication Administration Guide for Junos OS Evolved](#)

Chassis Features

SUMMARY

Learn where to find chassis-level software documentation for Junos OS Evolved.

IN THIS SECTION

- [General | 117](#)
- [Alarms, Defects, and Statistics | 117](#)
- [Internet Control Message Protocol \(ICMP\) | 117](#)
- [Precision Time Protocol \(PTP\) | 117](#)
- [Synchronous Ethernet \(SyncE\) | 118](#)
- [Related Guides | 118](#)

In general, Juniper features and technologies work the same on Junos OS and Junos OS Evolved, so much of the documentation applies to both operating systems. Given the incredible number of features and amount of documentation available, we want to help you find the most relevant content quickly.

Below we've collected recommended documentation topics and guides that apply to Junos OS and Junos OS Evolved. Alternatively, you can access these same lists from the [Junos OS Evolved Software Documentation](#) page.

NOTE: The documentation provided in this chapter doesn't apply to all Junos OS Evolved platforms and all Junos OS Evolved software releases. For a more detailed list of Junos OS Evolved platforms and the individual features each supports, see [Feature Explorer](#).

General

- [Check Overall CPU and Memory Usage](#)
- [Managing Errors](#)
- [request flight-recorder set high-cpu](#)
- [restart chassis-control](#)

Alarms, Defects, and Statistics

- [Alarms](#)
- [Interface Alarms](#)
- [Understanding Chassis Alarms](#)

Internet Control Message Protocol (ICMP)

- [ICMP Features](#)
- [Understanding the ICMP Protocol for Discovering Gateways to Other Networks](#)

Precision Time Protocol (PTP)

- [Precision Time Protocol](#)
- [PTP Clocks](#)

- [PTP Profiles](#)

Synchronous Ethernet (SyncE)

- [Ethernet Synchronization Message Channel \(ESMC\)](#)
- [Synchronous Ethernet](#)

Related Guides

- [Chassis-Level User Guide](#)
- [Time Management Administration Guide](#)
- [Transport and Internet Protocols User Guide](#)

Class of Service

SUMMARY

Learn where to find Class of Service (CoS) documentation for Junos OS Evolved.

IN THIS SECTION

- [General | 119](#)
- [Classification | 119](#)
- [Congestion Management | 119](#)
- [Forwarding Classes | 120](#)
- [Related Guides | 120](#)

In general, Juniper features and technologies work the same on Junos OS and Junos OS Evolved, so much of the documentation applies to both operating systems. Given the incredible number of features and amount of documentation available, we want to help you find the most relevant content quickly.

Below we've collected recommended documentation topics and guides that apply to Junos OS and Junos OS Evolved. Alternatively, you can access these same lists from the [Junos OS Evolved Software Documentation](#) page.

NOTE: The documentation provided in this chapter doesn't apply to all Junos OS Evolved platforms and all Junos OS Evolved software releases. For a more detailed list of Junos OS Evolved platforms and the individual features each supports, see [Feature Explorer](#).

General

- [Configuring Schedulers for Priority Scheduling](#)
- [Overriding the Input Classification](#)
- [Understanding Applying CoS Classifiers and Rewrite Rules to Interfaces](#)
- [Understanding CoS Forwarding Classes](#)

Classification

- [Assigning CoS Components to Interfaces](#)
- [Default IEEE 802.1p Classifier](#)
- [explicit-null-cos](#)
- [import](#)
- [traffic-manager](#)
- [Understanding CoS Classifiers](#)
- [Understanding CoS MPLS EXP Classifiers and Rewrite Rules](#)
- [Understanding How Behavior Aggregate Classifiers Prioritize Trusted Traffic](#)

Congestion Management

- [Managing Congestion Using RED Drop Profiles and Packet Loss Priorities](#)
- [Understanding CoS Explicit Congestion Notification](#)

- [Understanding CoS WRED Drop Profiles](#)

Forwarding Classes

- [Configuring a Custom Forwarding Class for Each Queue](#)
- [Configuring CoS-Based Forwarding](#)
- [Configuring Up to 16 Custom Forwarding Classes](#)

Related Guides

- [Class of Service User Guide \(Routers and EX9200 Switches\)](#)
- [Traffic Management User Guide \(QFX Series and EX4600 Switches\)](#)

Ethernet Switching and Bridging

SUMMARY

Learn where to find Ethernet switching and bridging documentation for Junos OS Evolved.

IN THIS SECTION

- [General | 121](#)
- [Data Center Bridging Technologies | 121](#)
- [Integrated Routing and Bridging \(IRB\) | 122](#)
- [Spanning Tree Protocol \(STP\) | 122](#)
- [VLANs | 122](#)
- [Related Guides | 122](#)

In general, Juniper features and technologies work the same on Junos OS and Junos OS Evolved, so much of the documentation applies to both operating systems. Given the incredible number of features and amount of documentation available, we want to help you find the most relevant content quickly.

Below we've collected recommended documentation topics and guides that apply to Junos OS and Junos OS Evolved. Alternatively, you can access these same lists from the [Junos OS Evolved Software Documentation](#) page.

NOTE: The documentation provided in this chapter doesn't apply to all Junos OS Evolved platforms and all Junos OS Evolved software releases. For a more detailed list of Junos OS Evolved platforms and the individual features each supports, see [Feature Explorer](#).

General

- [Configuring a Layer 2 Virtual Switch](#)
- [Configuring RSTP Protocol](#)
- [Configuring VSTP Protocol](#)
- [Device Discovery Using LLDP](#)
- [Layer 2 Networking](#)
- [native-vlan-id](#)
- [Understanding and Preventing Unknown Unicast Forwarding](#)
- [Understanding FCoE Transit Switch Functionality](#)

Data Center Bridging Technologies

- [Data Center Quantized Congestion Notification \(DCQCN\)](#)
- [Understanding CoS Hierarchical Port Scheduling \(ETS\)](#)
- [Understanding DCB Features and Requirements](#)
- [Understanding DCBX](#)
- [Understanding DCBX Application Protocol TLV Exchange](#)
- [Understanding PFC Using DSCP at Layer 3 for Untagged Traffic](#)
- [Understanding Priority-Based Flow Control](#)

Integrated Routing and Bridging (IRB)

- [Integrated Routing and Bridging](#)
- [Proxy ARP](#)

Spanning Tree Protocol (STP)

- [BPDU Protection for Spanning-Tree Protocols](#)
- [Configuring MSTP Protocol](#)
- [Loop Protection for Spanning-Tree Protocols](#)
- [Root Protection for VPLS Multihome Environments](#)
- [Spanning-Tree Protocol Overview](#)

VLANs

- [Bridging and VLANs](#)
- [Configuring Q-in-Q Tunneling and VLAN Q-in-Q Tunneling and VLAN Translation](#)
- [Configuring Tagged VLANs](#)
- [Enabling VLAN Tagging](#)
- [Layer 3 Logical Interfaces](#)
- [Multiple VLAN Registration Protocol](#)
- [Private VLANs](#)
- [Routed VLAN Interfaces](#)

Related Guides

- [Ethernet Switching User Guide](#)

- [Layer 2 Bridging, Address Learning, and Forwarding User Guide](#)

Forwarding Options

SUMMARY

Learn where to find forwarding options documentation for Junos OS Evolved.

IN THIS SECTION

- [Port Mirroring | 123](#)
- [Related Guides | 124](#)

In general, Juniper features and technologies work the same on Junos OS and Junos OS Evolved, so much of the documentation applies to both operating systems. Given the incredible number of features and amount of documentation available, we want to help you find the most relevant content quickly.

Below we've collected recommended documentation topics and guides that apply to Junos OS and Junos OS Evolved. Alternatively, you can access these same lists from the [Junos OS Evolved Software Documentation](#) page.

NOTE: The documentation provided in this chapter doesn't apply to all Junos OS Evolved platforms and all Junos OS Evolved software releases. For a more detailed list of Junos OS Evolved platforms and the individual features each supports, see [Feature Explorer](#).

Port Mirroring

- [Configuring Port Mirroring](#)
- [Configuring Port Mirroring and Analyzers](#)
- [Configuring Port Mirroring for Remote Destinations](#)
- [Configuring Port Mirroring Local and Remote Analysis](#)
- [Configuring Port Mirroring on Logical Interfaces](#)
- [Configuring Port Mirroring on Physical Interfaces](#)

- [Configuring Traffic Sampling](#)
- [Port Mirroring and Analyzers](#)

Related Guides

- [Monitoring, Sampling, and Collection Services Interfaces User Guide](#)
- [Network Management and Monitoring Guide](#)
- [Traffic Sampling, Forwarding, and Monitoring User Guide](#)

High Availability

SUMMARY

Learn where to find high availability documentation for Junos OS Evolved.

IN THIS SECTION

- [General](#) | **125**
- [Graceful Restart](#) | **125**
- [Link Aggregation Groups \(LAGs\)](#) | **125**
- [Load Balancing](#) | **126**
- [Virtual Router Redundancy Protocol \(VRRP\)](#) | **126**
- [Related Guides](#) | **126**

In general, Juniper features and technologies work the same on Junos OS and Junos OS Evolved, so much of the documentation applies to both operating systems. Given the incredible number of features and amount of documentation available, we want to help you find the most relevant content quickly.

Below we've collected recommended documentation topics and guides that apply to Junos OS and Junos OS Evolved. Alternatively, you can access these same lists from the [Junos OS Evolved Software Documentation](#) page.

NOTE: The documentation provided in this chapter doesn't apply to all Junos OS Evolved platforms and all Junos OS Evolved software releases. For a more detailed list of Junos OS Evolved platforms and the individual features each supports, see [Feature Explorer](#).

General

- [Configuring Nonstop Active Routing](#)
- [Graceful Routing Engine Switchover System Requirements](#)
- [Nonstop Active Routing System Requirements](#)
- [Understanding Graceful Routing Engine Switchover](#)
- [Understanding Routing Engine Redundancy](#)

Graceful Restart

- [Configuring Graceful Restart for MPLS-Related Protocols](#)
- [Configuring Graceful Restart for Routing Protocols](#)
- [Configuring Graceful Restart for VPNs](#)
- [graceful-restart \(Protocols BGP\)](#)
- [Understanding Graceful Restart for BGP](#)
- [Understanding PIM Sparse Mode](#)

Link Aggregation Groups (LAGs)

- [Aggregated Ethernet Interfaces](#)
- [Aggregated Ethernet Interfaces](#)
- [Load Balancing on Aggregated Ethernet Interfaces](#)

Load Balancing

- [Load Balancing for a BGP Session](#)
- [maximum-ecmp](#)
- [Resilient Hashing on LAGs and ECMP groups](#)
- [Understanding the Algorithm Used to Load Balance Traffic on MX Series Routers](#)

Virtual Router Redundancy Protocol (VRRP)

- [Junos OS Support for VRRPv3](#)
- [Understanding VRRP](#)
- [Understanding VRRP](#)

Related Guides

- [BGP User Guide](#)
- [Ethernet Interfaces User Guide for Routing Devices](#)
- [High Availability User Guide](#)
- [Interfaces User Guide for Switches](#)

Interfaces

SUMMARY

Learn where to find interfaces documentation for Junos OS Evolved.

IN THIS SECTION

- [General](#) | 127

- [Address Resolution Protocol \(ARP\) | 128](#)
- [Aggregated Ethernet interface | 128](#)
- [Flexible Tunnel Interfaces \(FTI\) | 128](#)
- [Logical Interfaces \(IFL\) and Physical Interfaces \(IFD\) | 128](#)
- [Related Guides | 129](#)

In general, Juniper features and technologies work the same on Junos OS and Junos OS Evolved, so much of the documentation applies to both operating systems. Given the incredible number of features and amount of documentation available, we want to help you find the most relevant content quickly.

Below we've collected recommended documentation topics and guides that apply to Junos OS and Junos OS Evolved. Alternatively, you can access these same lists from the [Junos OS Evolved Software Documentation](#) page.

NOTE: The documentation provided in this chapter doesn't apply to all Junos OS Evolved platforms and all Junos OS Evolved software releases. For a more detailed list of Junos OS Evolved platforms and the individual features each supports, see [Feature Explorer](#).

General

- [Configuring Accounting Options, Source Class Usage and Destination Class Usage Options](#)
- [Flow Control for Ethernet Interfaces](#)
- [Generic Routing Encapsulation \(GRE\)](#)
- [Gigabit Ethernet Autonegotiation](#)
- [Introduction to OAM Link Fault Management \(LFM\)](#)
- [Layer 3 Logical Interfaces](#)
- [Managing Errors](#)
- [Next-Hop-Based Dynamic Tunnels](#)
- [show | display set](#)

- [show interfaces diagnostics optics](#)

Address Resolution Protocol (ARP)

- [ARP](#)
- [Configuring Static ARP Table Entries For Mapping IP Addresses to MAC Addresses](#)
- [Proxy ARP](#)
- [show arp](#)

Aggregated Ethernet interface

- [Link Protection of Aggregated Ethernet Interfaces](#)
- [Load Balancing for Aggregated Ethernet Interfaces](#)
- [maximum-links](#)
- [periodic](#)
- [Physical Interface Properties](#)

Flexible Tunnel Interfaces (FTI)

- [Configuring Flexible Tunnel Interfaces](#)
- [Configuring IP Tunnel Interfaces](#)

Logical Interfaces (IFL) and Physical Interfaces (IFD)

- [Configuring a Logical Interface for Trunk Mode](#)
- [Configuring Dual VLAN Tags](#)
- [Configuring Tagged VLANs](#)

- [Damping Physical Interfaces](#)
- [Flexible Ethernet Services Encapsulation](#)
- [flexible-vlan-tagging](#)
- [Forward Error Correction \(FEC\) and Bit Error Rate \(BER\)](#)
- [Interface Ranges for Physical Interfaces](#)
- [Physical Interface Properties](#)
- [Protocol Family and Interface Address Properties](#)
- [vlan-tags \(Stacked VLAN Tags\)](#)

Related Guides

- [Ethernet Interfaces User Guide for Routing Devices](#)
- [Ethernet Switching User Guide](#)
- [Getting Started with Junos OS Evolved](#)
- [Interfaces Fundamentals for Junos OS Evolved](#)
- [Interfaces User Guide for Switches](#)

IPv6

SUMMARY

Learn where to find IPv6 documentation for Junos OS Evolved.

IN THIS SECTION

- [General | 130](#)
- [Related Guides | 130](#)

In general, Juniper features and technologies work the same on Junos OS and Junos OS Evolved, so much of the documentation applies to both operating systems. Given the incredible number of features and amount of documentation available, we want to help you find the most relevant content quickly.

Below we've collected recommended documentation topics and guides that apply to Junos OS and Junos OS Evolved. Alternatively, you can access these same lists from the [Junos OS Evolved Software Documentation](#) page.

NOTE: The documentation provided in this chapter doesn't apply to all Junos OS Evolved platforms and all Junos OS Evolved software releases. For a more detailed list of Junos OS Evolved platforms and the individual features each supports, see [Feature Explorer](#).

General

- [Configure Static Routes](#)
- [Configuring Junos OS Routing Tables](#)
- [Filter-Based Forwarding Overview](#)
- [Introduction to OSPF](#)
- [IPv6 Multicast Flow](#)
- [IPv6 Neighbor Discovery](#)
- [IPv6 Overview](#)
- [IPv6 Traffic over Layer 3 VPNs](#)
- [IPv6-over-Ipv4 Tunnels](#)
- [ipv6-tunneling](#)
- [label-switched-path \(Protocols MPLS\)](#)
- [Neighbor Discovery Cache Protection](#)
- [Overview of System Logging](#)
- [Path MTU Discovery](#)

Related Guides

- [IPv6 Neighbor Discovery User Guide](#)

- [MPLS Applications User Guide](#)

Multicast

SUMMARY

Learn where to find multicast documentation for Junos OS Evolved.

IN THIS SECTION

- [General | 131](#)
- [IGMP Snooping | 132](#)
- [Internet Group Management Protocol \(IGMP\) | 132](#)
- [Protocol Independent Multicast \(PIM\) | 132](#)
- [Related Guide | 133](#)

In general, Juniper features and technologies work the same on Junos OS and Junos OS Evolved, so much of the documentation applies to both operating systems. Given the incredible number of features and amount of documentation available, we want to help you find the most relevant content quickly.

Below we've collected recommended documentation topics and guides that apply to Junos OS and Junos OS Evolved. Alternatively, you can access these same lists from the [Junos OS Evolved Software Documentation](#) page.

NOTE: The documentation provided in this chapter doesn't apply to all Junos OS Evolved platforms and all Junos OS Evolved software releases. For a more detailed list of Junos OS Evolved platforms and the individual features each supports, see [Feature Explorer](#).

General

- [Configuring MLD](#)
- [Examples: Configuring Bandwidth Management](#)
- [Examples: Configuring MSDP](#)

- [Understanding Unicast RPF \(Routers\)](#)

IGMP Snooping

- [Configure Multicast Forwarding with IGMP Snooping in an EVPN-MPLS Environment](#)
- [IGMP Snooping Overview](#)
- [Overview of Multicast Forwarding with IGMP Snooping or MLD Snooping in an EVPN-VXLAN Environment](#)

Internet Group Management Protocol (IGMP)

- [Configuring IGMP](#)
- [Multicast Overview](#)

Protocol Independent Multicast (PIM)

- [Configuring PIM Bootstrap Router](#)
- [Configuring PIM Dense Mode](#)
- [Configuring PIM Filtering](#)
- [Configuring PIM Join Load Balancing](#)
- [Configuring Static RP](#)
- [Example: Configuring Anycast RP](#)
- [Example: Configuring Bidirectional PIM](#)
- [Examples: Configuring PIM RPT and SPT Cutover](#)
- [Examples: Configuring Reverse Path Forwarding](#)
- [IPv6 Multicast Flow](#)
- [PIM Overview](#)

- [Understanding PIM Auto-RP](#)
- [Understanding PIM Sparse Mode](#)

Related Guide

[Multicast Protocols User Guide](#)

Multiprotocol Label Switching Applications

SUMMARY

Learn where to find Multiprotocol Label Switching (MPLS) documentation for Junos OS Evolved.

IN THIS SECTION

- [General | 134](#)
- [Label Distribution Protocol \(LDP\) | 134](#)
- [Label-Switched Paths \(LSPs\) | 134](#)
- [Static LSPs | 134](#)
- [Traffic Engineering | 135](#)
- [Related Guides | 135](#)

In general, Juniper features and technologies work the same on Junos OS and Junos OS Evolved, so much of the documentation applies to both operating systems. Given the incredible number of features and amount of documentation available, we want to help you find the most relevant content quickly.

Below we've collected recommended documentation topics and guides that apply to Junos OS and Junos OS Evolved. Alternatively, you can access these same lists from the [Junos OS Evolved Software Documentation](#) page.

NOTE: The documentation provided in this chapter doesn't apply to all Junos OS Evolved platforms and all Junos OS Evolved software releases. For a more detailed list of Junos OS Evolved platforms and the individual features each supports, see [Feature Explorer](#).

General

- [Load Balancing MPLS Traffic](#)
- [LSP Routes](#)
- [MPLS Overview](#)
- [PCEP Configuration](#)
- [Point-to-Multipoint LSP Configuration](#)
- [Supported PCEP Standards](#)

Label Distribution Protocol (LDP)

- [LDP Configuration](#)
- [LDP Overview](#)

Label-Switched Paths (LSPs)

- [Basic LSP Configuration](#)
- [label-switched-path \(Protocols MPLS\)](#)
- [Link Protection for MPLS LSPs](#)
- [Node and Path Protection for MPLS LSPs](#)
- [Using MPLS to Diagnose LSPs, VPNs, and Layer 2 Circuits](#)

Static LSPs

- [Primary, Secondary, and Static LSP Configuration](#)
- [Segment Routing LSP Configuration](#)

Traffic Engineering

- [DiffServ-Aware Traffic Engineering Configuration](#)
- [IS-IS Extensions to Support Traffic Engineering](#)
- [MPLS Traffic Engineering Configuration](#)
- [RSVP Configuration](#)
- [RSVP Overview](#)

Related Guides

- [MPLS Applications User Guide](#)
- [Standards Reference](#)

Network Management and Monitoring

SUMMARY

Learn where to find network management and monitoring documentation for Junos OS Evolved.

IN THIS SECTION

- [General](#) | **136**
- [Junos XML Management Protocol](#) | **136**
- [MPLS Operations, Administration, and Maintenance \(OAM\)](#) | **137**
- [NETCONF](#) | **137**
- [Simple Network Management Protocol \(SNMP\)](#) | **137**
- [Simple Network Management Protocol \(SNMP\) Management Information Base \(MIB\) and Traps](#) | **138**
- [Related Guides](#) | **138**

In general, Juniper features and technologies work the same on Junos OS and Junos OS Evolved, so much of the documentation applies to both operating systems. Given the incredible number of features and amount of documentation available, we want to help you find the most relevant content quickly.

Below we've collected recommended documentation topics and guides that apply to Junos OS and Junos OS Evolved. Alternatively, you can access these same lists from the [Junos OS Evolved Software Documentation](#) page.

NOTE: The documentation provided in this chapter doesn't apply to all Junos OS Evolved platforms and all Junos OS Evolved software releases. For a more detailed list of Junos OS Evolved platforms and the individual features each supports, see [Feature Explorer](#).

General

- [Configuring Connectivity Fault Management \(CFM\)](#)
- [Introduction to OAM Connectivity Fault Management \(CFM\)](#)
- [Management Interface in a Non-Default Instance](#)
- [sFlow Monitoring Technology](#)
- [traceroute](#)
- [Tracing Architecture](#)

Junos XML Management Protocol

- [Commit and Synchronize Ephemeral Configuration Data Using the NETCONF or Junos XML Protocol](#)
- [Junos XML Management Protocol and Junos XML API Overview](#)
- [Rename Objects In Configuration Data Using the Junos XML Protocol](#)
- [Reorder Elements In Configuration Data Using the Junos XML Protocol](#)
- [Replace Patterns in Configuration Data Using the NETCONF or Junos XML Protocol](#)
- [Specify the Output Format for Operational Information Requests in a NETCONF Session](#)

MPLS Operations, Administration, and Maintenance (OAM)

- [explicit-null \(Protocols MPLS\)](#)
- [traceroute mpls ldp](#)
- [ultimate-hop-popping](#)
- [Using MPLS to Diagnose LSPs, VPNs, and Layer 2 Circuits](#)

NETCONF

- [Configure RFC-Compliant NETCONF Sessions](#)
- [Establish an SSH Connection for a NETCONF Session](#)
- [NETCONF and Shell Sessions over Outbound HTTPS](#)
- [NETCONF Java Toolkit Overview](#)
- [NETCONF XML Management Protocol Developer Guide](#)
- [OpenConfig Overview](#)
- [Understanding Junos OS YANG Modules](#)
- [Understanding Python Automation Scripts for Devices Running Junos OS](#)
- [Understanding the YANG Modules That Define the Junos OS Configuration](#)

Simple Network Management Protocol (SNMP)

- [Network Monitoring by using SNMP](#)
- [Remote Network Monitoring \(RMON\)](#)
- [SNMP MIB Explorer](#)

Simple Network Management Protocol (SNMP) Management Information Base (MIB) and Traps

- [Configuring Basic SNMP](#)
- [Network Monitoring by using SNMP](#)
- [SNMP MIBs and Traps Supported by Junos OS](#)
- [SNMP Scripts Overview](#)

Related Guides

- [gRPC Network Services User Guide](#)
- [Junos XML Management Protocol Developer Guide](#)
- [MPLS Applications User Guide](#)
- [Network Management and Monitoring Guide](#)

Network Security

SUMMARY

Learn where to find network security documentation for Junos OS Evolved.

IN THIS SECTION

- [General | 139](#)
- [Media Access Control Security \(MACsec\) | 139](#)
- [Public Key Infrastructure \(PKI\) | 139](#)
- [Related Guides | 140](#)

In general, Juniper features and technologies work the same on Junos OS and Junos OS Evolved, so much of the documentation applies to both operating systems. Given the incredible number of features and amount of documentation available, we want to help you find the most relevant content quickly.

Below we've collected recommended documentation topics and guides that apply to Junos OS and Junos OS Evolved. Alternatively, you can access these same lists from the [Junos OS Evolved Software Documentation](#) page.

NOTE: The documentation provided in this chapter doesn't apply to all Junos OS Evolved platforms and all Junos OS Evolved software releases. For a more detailed list of Junos OS Evolved platforms and the individual features each supports, see [Feature Explorer](#).

General

- [Control Plane Distributed Denial-of-Service \(DDoS\) Protection Overview](#)
- [Overview of Third-Party Applications on Junos OS Evolved](#)
- [request system malware-scan](#)

Media Access Control Security (MACsec)

- [cipher-suite \(MACsec\)](#)
- [Configuring Media Access Control Security \(MACsec\) on Routers](#)
- [mtu](#)
- [Understanding Media Access Control Security \(MACsec\)](#)

Public Key Infrastructure (PKI)

- [ca-profile \(Security PKI\)](#)
- [Certificate Authority](#)
- [Certificate Enrollment](#)
- [Certificate Revocation](#)
- [IPsec Basics](#)

- [proposal \(Security IPsec\)](#)
- [request security pki generate-key-pair \(Security\)](#)
- [Requesting a CA Digital Certificate](#)

Related Guides

- [IPsec VPN User Guide](#)
- [Security Services Administration Guide](#)

Routing Options

SUMMARY

Learn where to find routing options documentation for Junos OS Evolved.

IN THIS SECTION

- [General | 141](#)
- [Bidirectional Forwarding Detection \(BFD\) | 141](#)
- [Virtual Router Routing Instances | 142](#)
- [Virtual-Router \(VRF-Lite\) | 142](#)
- [Related Guides | 142](#)

In general, Juniper features and technologies work the same on Junos OS and Junos OS Evolved, so much of the documentation applies to both operating systems. Given the incredible number of features and amount of documentation available, we want to help you find the most relevant content quickly.

Below we've collected recommended documentation topics and guides that apply to Junos OS and Junos OS Evolved. Alternatively, you can access these same lists from the [Junos OS Evolved Software Documentation](#) page.

NOTE: The documentation provided in this chapter doesn't apply to all Junos OS Evolved platforms and all Junos OS Evolved software releases. For a more detailed list of Junos OS Evolved platforms and the individual features each supports, see [Feature Explorer](#).

General

- [BGP Monitoring Protocol](#)
- [Configure Static Routes](#)
- [Configuring Packet Forwarding Behavior](#)
- [Layer 2 Forwarding Tables](#)

Bidirectional Forwarding Detection (BFD)

- [Aggregated Ethernet Interfaces](#)
- [BFD for BGP Sessions](#)
- [Bidirectional Forwarding Detection \(BFD\) for MPLS](#)
- [Bidirectional Forwarding Detection for Static Routes](#)
- [Configuring PIM and the Bidirectional Forwarding Detection \(BFD\) Protocol](#)
- [Example: Configuring BFD for Static Routes for Faster Network Failure Detection](#)
- [Periodic Packet Management](#)
- [Understanding BFD for IS-IS](#)
- [Understanding BFD for OSPF](#)
- [Understanding BFD for Static Routes for Faster Network Failure Detection](#)
- [Understanding How BFD Detects Network Failures](#)

Virtual Router Routing Instances

- [Configuring Virtual-Router Routing Instances in VPNs](#)
- [Legacy DHCP and Extended DHCP](#)
- [Understanding Multicast Route Leaking for VRF and Virtual Router Instances](#)

Virtual-Router (VRF-Lite)

- [Configuring Virtual-Router Routing Instances in VPNs](#)
- [Example: Configuring Bidirectional PIM](#)

Related Guides

- [BGP User Guide](#)
- [Ethernet Interfaces User Guide for Routing Devices](#)
- [High Availability User Guide](#)
- [Multicast Protocols User Guide](#)
- [Protocol-Independent Routing Properties User Guide](#)

Routing Policies and Firewall Filters

SUMMARY

Learn where to find routing policies and firewall filters documentation for Junos OS Evolved.

IN THIS SECTION

- [General | 143](#)
- [Firewall Filters | 143](#)
- [Related Guides | 144](#)

In general, Juniper features and technologies work the same on Junos OS and Junos OS Evolved, so much of the documentation applies to both operating systems. Given the incredible number of features and amount of documentation available, we want to help you find the most relevant content quickly.

Below we've collected recommended documentation topics and guides that apply to Junos OS and Junos OS Evolved. Alternatively, you can access these same lists from the [Junos OS Evolved Software Documentation](#) page.

NOTE: The documentation provided in this chapter doesn't apply to all Junos OS Evolved platforms and all Junos OS Evolved software releases. For a more detailed list of Junos OS Evolved platforms and the individual features each supports, see [Feature Explorer](#).

General

- [Filter-Based Forwarding Overview](#)
- [legacy-redirect-ip-action](#)
- [Multiprotocol BGP](#)
- [policy-statement](#)
- [Routing Policy Overview](#)
- [Understanding Prepending AS Numbers to BGP AS Paths](#)

Firewall Filters

- [Configuring a Firewall Filter to De-Encapsulate GRE or IPIP Traffic](#)
- [Configuring Firewall Filters \(CLI Procedure\)](#)
- [Configuring Flexible Tunnel Interfaces](#)
- [Configuring Port Mirroring and Analyzers](#)
- [Example: Configuring a Firewall Filter on a Management Interface on an EX Series Switch](#)
- [Example: Transporting IPv6 Traffic Across IPv4 Using Filter-Based Tunneling](#)
- [Firewall Filter Flexible Match Conditions](#)

- [Firewall Filter Match Conditions and Actions \(QFX and EX Series Switches\)](#)
- [Firewall Filter Match Conditions Based on Bit-Field Values](#)
- [Firewall Filter Match Conditions for IPv4 Traffic](#)
- [Firewall Filter Match Conditions for IPv6 Traffic](#)
- [Firewall Filter Support on Loopback Interface](#)
- [Firewall Filters Overview](#)
- [Guidelines for Applying Standard Firewall Filters](#)
- [Guidelines for gRPC and gNMI Sensors \(Junos Telemetry Interface\)](#)
- [Interface-Specific Firewall Filter Instances Overview](#)
- [Overview of Firewall Filters \(QFX Series\)](#)
- [Planning the Number of Firewall Filters to Create](#)
- [Port Mirroring and Analyzers](#)
- [Troubleshooting Firewall Filter Configuration](#)
- [Understanding Firewall Filter Match Conditions](#)
- [Understanding Queuing and Marking of Host Outbound Traffic](#)
- [Understanding the Use of Policers in Firewall Filters](#)
- [Walkup for Route Filters Overview](#)

Related Guides

- [gRPC Network Services User Guide](#)
- [Interfaces Fundamentals for Junos OS Evolved](#)
- [Routing Policies, Firewall Filters, and Traffic Policers User Guide](#)

Routing Protocols

SUMMARY

Learn where to find routing protocols documentation for Junos OS Evolved.

IN THIS SECTION

- [General](#) | **145**
- [Border Gateway Protocol \(BGP\)](#) | **145**
- [Intermediate System-to-Intermediate System \(IS-IS\)](#) | **147**
- [Open Shortest Path First \(OSPF\)](#) | **147**
- [Related Guides](#) | **147**

In general, Juniper features and technologies work the same on Junos OS and Junos OS Evolved, so much of the documentation applies to both operating systems. Given the incredible number of features and amount of documentation available, we want to help you find the most relevant content quickly.

Below we've collected recommended documentation topics and guides that apply to Junos OS and Junos OS Evolved. Alternatively, you can access these same lists from the [Junos OS Evolved Software Documentation](#) page.

NOTE: The documentation provided in this chapter doesn't apply to all Junos OS Evolved platforms and all Junos OS Evolved software releases. For a more detailed list of Junos OS Evolved platforms and the individual features each supports, see [Feature Explorer](#).

General

[IPv6 Neighbor Discovery](#)

Border Gateway Protocol (BGP)

- [Autonomous Systems for BGP Sessions](#)
- [BGP 4-Byte AS Numbers](#)

- BGP Auto-Discovered Neighbors
- BGP Confederations for IBGP Scaling
- BGP Egress Traffic Engineering
- BGP Error Messages
- BGP MED Attribute
- BGP Monitoring Protocol
- BGP Origin Validation
- BGP Overview
- BGP Peering Sessions
- BGP PIC for Layer 3 VPNs
- BGP Route Authentication
- BGP Route Prioritization
- BGP Route Reflectors
- Example: Applying Routing Policies at Different Levels of the BGP Hierarchy
- Example: Configuring BGP Large Communities
- How BGP Communities and Extended Communities Are Evaluated in Routing Policy Match Conditions
- IP Security for BGP
- Link-State Distribution Using BGP
- Load Balancing for a BGP Session
- Local Preference for BGP Routes
- Mapping OpenConfig BGP Commands to Junos Configuration
- Multiprotocol BGP
- `show bgp summary`
- TCP Access Restriction for BGP
- Understanding BGP Communities, Extended Communities, and Large Communities as Routing Policy Match Conditions

- [Understanding Graceful Restart for BGP](#)

Intermediate System-to-Intermediate System (IS-IS)

- [How to Configure Multiple Independent IGP Instances of IS-IS](#)
- [LDP Configuration](#)
- [traffic-engineering \(Protocols IS-IS\)](#)
- [Understanding Adjacency Segments, Anycast Segments, and Configurable SRGB in SPRING](#)
- [Understanding Hitless Authentication Key Rollover for IS-IS](#)

Open Shortest Path First (OSPF)

- [Configuring Loop-Free Alternate Routes for OSPF](#)
- [Configuring OSPF Authentication](#)
- [Configuring OSPF Database Protection](#)
- [Configuring OSPF Route Control](#)
- [Configuring OSPF Routing Policy](#)
- [Configuring OSPF Support for Traffic Engineering](#)
- [How to Configure Flexible Algorithms in OSPF for Segment Routing Traffic Engineering](#)
- [Introduction to OSPF](#)
- [Understanding Adjacency Segments, Anycast Segments, and Configurable SRGB in SPRING](#)
- [Understanding Backup Selection Policy for OSPF Protocol](#)
- [Understanding LDP-IGP Synchronization](#)

Related Guides

- [BGP User Guide](#)

- [IS-IS User Guide](#)
- [MPLS Applications User Guide](#)
- [OSPF User Guide](#)
- [Routing Policies, Firewall Filters, and Traffic Policers User Guide](#)

Services Applications

SUMMARY

Learn where to find services applications documentation for Junos OS Evolved.

IN THIS SECTION

- [General](#) | **148**
- [Dynamic Host Configuration Protocol \(DHCP\)](#) | **149**
- [Flow Monitoring Services](#) | **149**
- [Related Guides](#) | **149**

In general, Juniper features and technologies work the same on Junos OS and Junos OS Evolved, so much of the documentation applies to both operating systems. Given the incredible number of features and amount of documentation available, we want to help you find the most relevant content quickly.

Below we've collected recommended documentation topics and guides that apply to Junos OS and Junos OS Evolved. Alternatively, you can access these same lists from the [Junos OS Evolved Software Documentation](#) page.

NOTE: The documentation provided in this chapter doesn't apply to all Junos OS Evolved platforms and all Junos OS Evolved software releases. For a more detailed list of Junos OS Evolved platforms and the individual features each supports, see [Feature Explorer](#).

General

- [Configure TWAMP on ACX and PTX Series Routers](#)

- [Understand Two-Way Active Measurement Protocol](#)
- [Understanding the REST API](#)
- [Understanding Using Probes for Real-Time Performance Monitoring on PTX Series Routers and QFX Switches](#)

Dynamic Host Configuration Protocol (DHCP)

- [DHCP Relay Agent Information Option \(Option 82\)](#)
- [DHCP Relay Agent over EVPN-VXLAN](#)
- [Legacy DHCP and Extended DHCP](#)
- [Secure DHCP Message Exchange](#)
- [user-defined-option-82](#)
- [Zero Touch Provisioning DHCP Options for Junos OS Evolved](#)

Flow Monitoring Services

- [Configuring Flow Aggregation to Use Version 9 Flow Templates](#)
- [Guidelines for gRPC and gNMI Sensors \(Junos Telemetry Interface\)](#)
- [Inline Active Flow Monitoring on IRB Interfaces](#)
- [Multiprotocol BGP](#)
- [sFlow Monitoring Technology](#)
- [Understand Inline Active Flow Monitoring](#)

Related Guides

- [gRPC Network Services User Guide](#)
- [Junos Telemetry Interface User Guide](#)

- [Monitoring, Sampling, and Collection Services Interfaces User Guide](#)
- [REST API Guide](#)

System Basics

SUMMARY

Learn where to find general system documentation for Junos OS Evolved.

IN THIS SECTION

- [Network Operating System | 150](#)
- [Related Guides | 151](#)

In general, Juniper features and technologies work the same on Junos OS and Junos OS Evolved, so much of the documentation applies to both operating systems. Given the incredible number of features and amount of documentation available, we want to help you find the most relevant content quickly.

Below we've collected recommended documentation topics and guides that apply to Junos OS and Junos OS Evolved. Alternatively, you can access these same lists from the [Junos OS Evolved Software Documentation](#) page.

NOTE: The documentation provided in this chapter doesn't apply to all Junos OS Evolved platforms and all Junos OS Evolved software releases. For a more detailed list of Junos OS Evolved platforms and the individual features each supports, see [Feature Explorer](#).

Network Operating System

- [How Junos OS Evolved Differs from Junos OS](#)
- [Junos OS Evolved Overview](#)
- [Top Differences Between Junos OS Evolved and Junos OS](#)

Related Guides

- [CLI User Guide for Junos OS Evolved](#)
- [Getting Started with Junos OS Evolved](#)
- [Introducing Junos OS Evolved](#)

System Management

SUMMARY

Learn where to find system management documentation for Junos OS Evolved.

IN THIS SECTION

- [General](#) | **152**
- [Juniper Extension Toolkit for Junos \(JET for Junos\)](#) | **152**
- [Scripts \(Op Scripts, Commit Scripts, Event Scripts\)](#) | **152**
- [Software Installation and Upgrade](#) | **153**
- [System Logging](#) | **153**
- [Related Guides](#) | **154**

In general, Juniper features and technologies work the same on Junos OS and Junos OS Evolved, so much of the documentation applies to both operating systems. Given the incredible number of features and amount of documentation available, we want to help you find the most relevant content quickly.

Below we've collected recommended documentation topics and guides that apply to Junos OS and Junos OS Evolved. Alternatively, you can access these same lists from the [Junos OS Evolved Software Documentation](#) page.

NOTE: The documentation provided in this chapter doesn't apply to all Junos OS Evolved platforms and all Junos OS Evolved software releases. For a more detailed list of Junos OS Evolved platforms and the individual features each supports, see [Feature Explorer](#).

General

- [Control Plane Distributed Denial-of-Service \(DDoS\) Protection Overview](#)
- [file copy](#)
- [Managing Licenses](#)
- [Managing Power](#)
- [Map Junos OS Command Output to JSON in the CLI](#)
- [ping](#)
- [Remote Access Overview](#)
- [request system snapshot \(Junos OS Evolved\)](#)
- [Root Password](#)
- [sFlow Monitoring Technology](#)
- [show log](#)
- [Understanding the REST API](#)
- [Validate the Configuration against the Installation Image](#)
- [Zero Touch Provisioning](#)

Juniper Extension Toolkit for Junos (JET for Junos)

- [JET Overview](#)
- [Juniper Extension Toolkit \(JET\)](#)
- [Overview of JET APIs](#)

Scripts (Op Scripts, Commit Scripts, Event Scripts)

- [Commit Script Overview](#)
- [Configure an Event Policy to Change the Configuration](#)

- [Event Policies and Event Notifications Overview](#)
- [Event Scripts Overview](#)
- [Execute an Op Script on the Local Device](#)
- [Global Parameters and Variables in Junos OS Automation Scripts](#)
- [How to Use the Requests Library for Python on Devices Running Junos OS](#)
- [Junos Automation Scripts Overview](#)
- [Op Script Overview](#)
- [Overview of Junos Automation Enhancements on Devices Running Junos OS with Enhanced Automation](#)
- [Overview of Python Modules on Devices Running Junos OS](#)
- [SLAX Overview](#)
- [SNMP Scripts Overview](#)
- [Understanding Python Automation Scripts for Devices Running Junos OS](#)

Software Installation and Upgrade

- [Junos OS Evolved Installation Packages](#)
- [Overview of Third-Party Applications on Junos OS Evolved](#)
- [Roll Back the Software to a Previous Version](#)

System Logging

- [Configuring System Logging for a Security Device](#)
- [Directing System Log Messages to a Remote Destination](#)
- [Overview of System Logging](#)
- [System Logging on a Single-Chassis System](#)

Related Guides

- [Automation Scripting User Guide](#)
- [CLI User Guide for Junos OS Evolved](#)
- [Juniper Extension Toolkit API Guide](#)
- [Junos OS Evolved Software Installation and Upgrade Guide](#)
- [Junos Snapshot Administrator in Python, Release 1.0](#)
- [Licensing Guide](#)
- [Network Management and Monitoring Guide](#)
- [User Access and Authentication Administration Guide for Junos OS Evolved](#)

Virtual Private Networks

SUMMARY

Learn where to find Virtual Private Networks (VPNs) documentation for Junos OS Evolved.

IN THIS SECTION

- [Ethernet Virtual Private Networks \(EVPN\) | 155](#)
- [L2 Circuit Cross Connect | 155](#)
- [Layer 2 Circuits | 156](#)
- [Layer 2 VPN \(L2 VPN\) | 156](#)
- [Layer 3 VPN \(L3 VPN\) | 156](#)
- [Related Guides | 156](#)

In general, Juniper features and technologies work the same on Junos OS and Junos OS Evolved, so much of the documentation applies to both operating systems. Given the incredible number of features and amount of documentation available, we want to help you find the most relevant content quickly.

Below we've collected recommended documentation topics and guides that apply to Junos OS and Junos OS Evolved. Alternatively, you can access these same lists from the [Junos OS Evolved Software Documentation](#) page.

NOTE: The documentation provided in this chapter doesn't apply to all Junos OS Evolved platforms and all Junos OS Evolved software releases. For a more detailed list of Junos OS Evolved platforms and the individual features each supports, see [Feature Explorer](#).

Ethernet Virtual Private Networks (EVPN)

- [EVPN Proxy ARP and ARP Suppression, and Proxy NDP and NDP Suppression](#)
- [EVPN with IRB Solution Overview](#)
- [Example: Configuring an ESI on a Logical Interface With EVPN Multihoming](#)
- [Example: Configuring Basic EVPN Active-Active Multihoming](#)
- [Examples: Tunneling Q-in-Q Traffic in an EVPN-VXLAN Overlay Network](#)
- [Symmetric Integrated Routing and Bridging with EVPN Type 2 Routes in EVPN-VXLAN Fabrics](#)
- [Understanding VLAN-Aware Bundle and VLAN-Based Service for EVPN](#)
- [Understanding When to Disable EVPN-VXLAN Core Isolation](#)
- [VLAN-Based Service for EVPN](#)

L2 Circuit Cross Connect

- [BFD Support for VCCV for Layer 2 VPNs, Layer 2 Circuits, and VPLS](#)
- [CCC, TCC, and Ethernet Over MPLS Configuration](#)
- [Configuring Local Interface Switching in Layer 2 Circuits](#)
- [Configuring Static Layer 2 Circuits](#)
- [local-switching \(Layer 2 Circuits\)](#)

Layer 2 Circuits

- [Aggregated Ethernet Interfaces](#)
- [input-vlan-map](#)
- [LDP Overview](#)

Layer 2 VPN (L2 VPN)

- [Example: Configure MPLS-Based Layer 2 VPNs](#)
- [Understanding FEC 129 BGP Autodiscovery for VPWS](#)

Layer 3 VPN (L3 VPN)

- [Carrier-of-Carrier VPNs](#)
- [Creating Unique VPN Routes Using VRF Tables](#)
- [Distributing VPN Routes](#)
- [Example: Configure a Basic MPLS-Based Layer 3 VPN](#)
- [Improving Layer 3 VPN Performance](#)
- [Interprovider VPNs](#)
- [Overview](#)
- [Virtual Routing Instances](#)
- [vrf-table-label](#)

Related Guides

- [EVPN User Guide](#)
- [Layer 2 VPNs and VPLS User Guide for Routing Devices](#)

- [Layer 3 VPNs User Guide for Routing Devices](#)

5

CHAPTER

Operational Commands

`show nlsd mode` | 159

`show nsagentd (statistics | status)` | 160

show nlsd mode

IN THIS SECTION

- [Syntax | 159](#)
- [Description | 159](#)
- [Additional Information | 159](#)
- [Required Privilege Level | 159](#)
- [Sample Output | 160](#)
- [Release Information | 160](#)

Syntax

```
show nlsd mode
```

Description

The command `show nlsd mode` shows you if the netlink asynchronous (async) API is activated or deactivated.

Additional Information

Netlink async mode is deactivated by default. You can enable netlink async mode with the command `set system netlink-async-mode`.

Required Privilege Level

view

Sample Output

show nlsd mode

```
user@host> show nlsd mode
Netlink async mode status: activated
```

Release Information

Command introduced in Junos OS Evolved Release 20.1R1.

show nsagentd (statistics | status)

IN THIS SECTION

- [Syntax | 160](#)
- [Description | 161](#)
- [Required Privilege Level | 161](#)
- [Sample Output | 161](#)
- [Release Information | 162](#)

Syntax

```
show nsagentd statistics
```

```
show nsagentd status
```

Description

The `nsagentd` options show you the statistics and status of the `nsagentd` app. This app handles and resolves TCP-specific functionality, updates, and notifications when a TCP flow becomes stale.

Required Privilege Level

view

Sample Output

show nsagentd statistics

```
user@host> show nsagentd statistics
Invalid upcalls:          0
Ingress SYN upcalls:
  Upcalls:                0
  FIB lookups:
    Attempted:            0
    Failed:               0
    Succeeded:            0
    Result code not EOK:  0
    Result code EOK:      0
  Ioctl invocations:      0
    Attempted:            0
    Failed:               0
    Succeeded:            0
  Stale TCP session upcalls:
    Upcalls:              0
    FIB lookups:
      Attempted:          0
      Failed:             0
      Succeeded:          0
      Result code not EOK: 0
      Result code EOK:    0
    Ioctl invocations:
      Attempted:          0
```



```
Failed:          0
Succeeded:       0
```

show nsagentd status

```
user@host> show nsagentd status
Local address:    224.0.0.4
Start time:       2022-04-06 08:05:27.535
Last upcall time: Never
```

Release Information

Command introduced in Junos OS Evolved Release 21.3R1.