

Introducing Junos OS Evolved

Published
2021-12-16

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 © 2021 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 | v

1

Overview of Junos OS Evolved

Junos OS Evolved Overview | 2

How Junos OS Evolved Differs from Junos OS | 4

Default Directories for Junos OS Evolved File Storage | 34

Junos OS Evolved Components and Processes | 35

Shell Commands for Junos OS Evolved | 39

Where to Find Information on Common Procedures | 41

2

Junos OS Evolved Configuration Overview

Junos OS Evolved Configuration Basics | 43

Methods for Configuring Junos OS Evolved | 43

Junos OS Evolved Configuration from External Devices | 46

3

Running 3rd Party Applications with Junos OS Evolved

Overview of Third-Party Applications on Junos OS Evolved | 48

Introduction to Third-Party Applications on Junos OS Evolved | 48

Running Applications in Containers vs Using Signing Keys | 48

Applications in Containers | 49

Using Signing Keys | 49

Security Caveats | 50

Application Pre-requisites | 50

Application APIs | 50

How to Run Applications | 51

Using Intercept Libraries | 52

Running Third-Party Applications in Containers | 61

- Deploying a Docker Container | 62
- Managing a Docker Container | 63
- Enabling Netlink or Packet IO in a Container | 63
- Selecting a VRF for a Docker Container | 64
- Modifying Resource Limits for Containers | 64

Protecting the Integrity of Junos OS Evolved with IMA | 65

Signing Third-Party Applications to Run Natively on Junos OS Evolved | 66

- Signing Keys Overview | 67
- Generating Signing Keys | 67
- Importing Signing Keys into the System Keystore and IMA Extended Keyring | 69
- Viewing the System Keystore and IMA Extended Keyring | 70
- How to Sign Applications | 71
- How to Run Signed Applications | 72

Removing Third-Party Applications | 72

Running Third-Party Applications in Containers | 73

- Deploying a Docker Container | 74
- Managing a Docker Container | 75
- Enabling Netlink or Packet IO in a Container | 76
- Selecting a VRF for a Docker Container | 76
- Modifying Resource Limits for Containers | 77

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](#)

[How Junos OS Evolved Differs from Junos OS | 4](#)

[Default Directories for Junos OS Evolved File Storage | 34](#)

[Junos OS Evolved Components and Processes | 35](#)

[Shell Commands for Junos OS Evolved | 39](#)

[Where to Find Information on Common Procedures | 41](#)

Junos OS Evolved Overview

IN THIS SECTION

- [Benefits | 2](#)
- [Native Linux Base | 3](#)
- [Central 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.

Central 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 central database called the Distributed Data Store (DDS).

The DDS does not interpret state. Its only job is to hold state received from subscribers and propagate state to consumers. It implements the publish-subscribe messaging pattern for communicating state between applications that are originators of a state to applications that are consumers of that state (see

- [Modified CLI Statements and Commands \(Junos OS Evolved\) | 16](#)
- [Changed CLI Command Output \(Junos OS Evolved\) | 23](#)
- [Removed CLI Statements and Commands \(Junos OS Evolved\) | 27](#)
- [XML Differences Between Junos OS and Junos OS Evolved | 30](#)

In many ways, Junos OS Evolved is the same as Junos OS: Key applications such as the routing, bridging, and management software is the same in both. And management plane interfaces and APIs, such as CLI, NETCONF, JET, JTI, AFI, and underlying data models, remain highly consistent. There are, however, some differences in behavior, 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.

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 1 on page 5](#).

Table 1: How Junos OS Evolved Behavior Differs from Junos OS

Junos OS Evolved Behavior	Junos OS Behavior	Link to Documentation
Access and Authorization		
You must set up the password-less login between two devices to use <code>jcs:open</code> 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>
Interfaces		

Table 1: How Junos OS Evolved Behavior Differs from Junos OS *(Continued)*

Junos OS Evolved Behavior	Junos OS Behavior	Link to Documentation
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>Installing the Software Package on a Router with a Single Routing Engine</i>
The management interface name format changed to re0:mgmt-0/re0: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 fxp0, and some use em0.	<i>Understanding Management Ethernet Interfaces</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 [edit system syslog host <i>ip-address</i>] hierarchy, you only need to configure the management-instance statement to use the mgmt_junos routing instance. You do not need to configure the mgmt_junos routing instance at the [edit system syslog host <i>ip-address</i> routing-instance] hierarchy.	Configure the mgmt_junos routing instance at the [edit system syslog host <i>ip-address</i> routing-instance] hierarchy if you want to send syslog messages to a remote host that is identified by its IP address at the [edit system syslog host <i>ip-address</i>] hierarchy.	<i>routing-instance</i>
When you issue the show firewall filter ? 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 show firewall application routing command.	When you issue the show firewall filter ? 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>

Table 1: How Junos OS Evolved Behavior Differs from Junos OS *(Continued)*

Junos OS Evolved Behavior	Junos OS Behavior	Link to Documentation
<p>In an untagged LAG, child 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>Configuring Aggregated Ethernet Interfaces on PTX Series Packet Transport Routers</p>
<p>When a new interface is added as a member to an AE bundle, the new member interface flaps: the physical interface is deleted as a regular interface and then added back in as an AE member and the statistics are reset.</p>	<p>When a new interface is added as a member to an AE 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>Configuration Guidelines for Aggregated Ethernet Interfaces and <i>Understanding Aggregated Ethernet Interfaces and LACP for Switches</i></p>
<p>For Junos OS Evolved, the software 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>For Junos OS, there is a limit of 64 member (or child) interfaces in an aggregated interface.</p>	<p>Configuration Guidelines for Aggregated Ethernet Interfaces and <i>Understanding Aggregated Ethernet Interfaces and LACP for Switches</i></p>
<p>In Junos OS Evolved, when you configure set interfaces <i>interface</i> ether-options 802.3ad <i>ae name</i> at the same time as you apply a second configuration to the same interface at the [edit interfaces <i>interface</i>] hierarchy, the second configuration will not take effect until the interface joins the aggregated Ethernet interface <i>ae name</i>. This is also true for gigether-options, when supported.</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> gigether-options] hierarchies and will be effective when applied.</p>	<p><i>ether-options</i>, <i>gigether-options</i></p>

Table 1: How Junos OS Evolved Behavior Differs from Junos OS *(Continued)*

Junos OS Evolved Behavior	Junos OS Behavior	Link to Documentation
<p>Starting from Junos OS Evolved Release 21.1R1, we changed the default FEC for 25-Gigabit and 50-Gigabit 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>	<p>In Junos OS, you can configure forward error correction (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.</p> <p>The default for 25-Gigabit and 50-Gigabit in Junos is FEC74.</p>	<i>fec (gigether)</i>
On PTX10004 and PTX10008 platforms running Junos OS Evolved, GRES is enabled by default and cannot be disabled.	In Junos OS, GRES is disabled by default.	<i>Understanding Graceful Routing Engine Switchover</i>

Messaging

The process eventd does not give any warning message if there are duplicate policies. Instead eventd accepts the policy on a first-come, first-served basis.	The process eventd gives a warning message if you try to create duplicate policies.	<i>Event Policies and Event Notifications Overview</i>
When the regular expression is to return empty matches, no error message is displayed.	When the regular expression is to return empty matches, you get the following error: regex error: empty (sub)expression	<i>Junos System Log Regular Expression Operators for the match Statement</i>
For op scripts run with the max-datasize configuration statement configured for the minimum, an error occurs. In Junos OS Evolved, the error is "Out of memory."	For op scripts run with the max-datasize configuration statement configured for the minimum, an error occurs. In Junos OS, the error is "Memory allocation failed."	<i>max-datasize</i>

Table 1: 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 RE. Backup RE messages are found in the messages file on the primary RE.	The messages file is written on both the primary RE and the backup RE.	<i>Displaying System Log Files</i>
Troubleshooting		
For Junos OS Evolved, a core file created during early bootup is stored in /var/core/re . But a core 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 <code>show system core-dumps</code> continues to show all cores generated.	For Junos OS, cores files are stored in /var/crash or /var/tmp .	<i>show system core-dumps</i>
The <code>request system snapshot</code> command takes a snapshot of the contents of the /soft directory only.	The <code>request system snapshot</code> command takes a snapshot of the contents of the /var/log , /var/core , /var/tmp , and /soft directories.	<i>request system snapshot</i> , Back Up the Currently Running and Active File System , <i>Understanding How to Back Up an Installation on Switches</i>
The hierarchy <code>set system scripts commit traceoptions</code> does not exist. <code>traceoptions</code> is disabled for <code>op</code> , <code>event</code> , and <code>commit</code> scripts.	Use <code>traceoptions</code> to define tracing operations that track traffic flow or routing protocol functionality in the routing device.	<i>traceoptions</i>
Junos OS Evolved stores the trace data for all scripts under the <code>cscript</code> application. 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>Tracing Script Processing on Devices Running Junos OS Evolved</i>

Table 1: How Junos OS Evolved Behavior Differs from Junos OS (Continued)

Junos OS Evolved Behavior	Junos OS Behavior	Link to Documentation
The request system storage cleanup command does not remove Junos OS Evolved images from the device. It removes all core files, log files from <code>/var/log/</code> , and all <code>/var/log/*</code> files. To remove old images from the device, use the <code>request system software delete</code> 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 <code>/var/crash</code> , log files from <code>/var/log/</code> , and certain other files from <code>/var/tmp</code> .	<i>request system storage cleanup</i>
User Interface		
The menu used for root password recovery is the Grub Menu. *Primary ptx-fixed-19.1-16 Primary [Recover password] Primary-Rollback ptx-fixed-19.1-15 Primary-Rollback [Recover password]	The menu used for root password recovery in Junos OS is the Junos Main Menu (the Recovery mode option).	<i>Recovering Root Password</i>
The show system firmware 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 show system firmware command appears as OK. But the fault is visible with the commands show chassis alarms, show chassis fpc, and so on.	When the FRU is offline, the cached firmware information of the FRU is not available to see.	<i>show system firmware</i>

New CLI Statements and Commands (Junos OS Evolved)

The changes in infrastructure between Junos OS and Junos OS Evolved sometimes require different CLI configuration statements and operational commands. For example, there is a new hierarchy level of statements in Junos OS Evolved that are not in Junos OS: `[edit security host-vpn]`. For more on these new statements and commands, see [Table 2 on page 11](#).

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

Statement or Command	Description	Link
New Statements		
[edit system extensions extension- service application file <i>filename</i> interpreter (bash python python3)]	You can specify whether a device running Junos OS Evolved should run a daemonized on-device JET application using Bash, Python, or Python 3.	<i>file</i>
[edit services monitoring twamp]	<p>Starting in Junos OS Evolved Release 20.3R1, you can configure the TWAMP monitoring service on PTX10003 routers. This service sends out probes to measure network performance. TWAMP is often used to check compliance with service-level agreements. The support for this service is limited to the following:</p> <ul style="list-style-type: none"> • IPv4 traffic only 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 only • Unauthenticated mode only 	<i>Understanding Two-Way Active Measurement Protocol on Routers and twamp</i>

Table 2: New CLI Statements and Commands (Junos OS Evolved) *(Continued)*

Statement or Command	Description	Link
[edit security host-vpn]	Support for host IPsec in the control plane only (that is, IPsec between the router and external management devices, which is not available in Junos OS. This statement configures a host-to-host VPN type of IPsec connection. Use the connections, ike-log, and ike-secrets statements at the [edit security host-vpn] hierarchy level to configure IKE and IPsec values.	<i>Overview of IPsec and host-vpn</i>
[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, remote, 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 know if a peer has gone dead.	<i>connections (Host VPN)</i>
[edit security host-vpn ike-log]	Statements at the [edit security host-vpn] hierarchy level used to configure IKE and IPsec values.	<i>ike-log</i>
[edit security host-vpn ike-secrets]	Statements at the [edit security host-vpn] hierarchy level used to configure IKE and IPsec values.	<i>ike-secrets</i>
[edit security host-vpn remote]	Configure identity details for authenticating the remote device during IKE negotiations.	<i>remote (Host VPN)</i>

Table 2: New CLI Statements and Commands (Junos OS Evolved) *(Continued)*

Statement or Command	Description	Link
[edit system trace application]	For Junos OS Evolved, trace data from all applications on all nodes is collected on the Routing Engine. You can view collected traces with the <code>show trace</code> command. You can remove inactive tracing sessions with the <code>clear trace</code> command.	<i>trace</i>
New Commands		
<code>clear security host-vpn security-associations</code>	Clear host IPsec security association information. You can configure host IPsec with the [edit security host-vpn] statement.	<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.	<i>clear trace</i>
<code>request services monitoring twamp client</code>	Start or stop a Two-Way Active Measurement Protocol (TWAMP) session.	<i>request services monitoring twamp client</i>
<code>request system application</code>	Start a specific process (for example <code>cmdd</code>) on the node you specify.	<i>request system application</i>
<code>request system debug-info</code>	Collect debug information from Junos OS Evolved, such as logs. The logs are stored in the <code>/var/tmp/debug_collector_timestamp</code> directory. Use the <code>node</code> option to collect information from a specific node.	<i>request system debug-info</i>

Table 2: New CLI Statements and Commands (Junos OS Evolved) *(Continued)*

Statement or Command	Description	Link
<code>request system software validate-restart</code>	The command performs a dry run of the <code>request system software add restart</code> command and displays the ISSU impact of the new restart option. See <i>request system software add</i> for more on the restart option.	<i>request system software validate-restart</i>
<code>restart app-name</code>	The following message will be logged when the restart command is used: App restarting <app name>. Related apps that may be impacted - <related-app name> .	<i>restart</i>
<code>show chassis routing-engine hard-disk-test</code>	Display the health of the hard disk. Use <code>disk /dev/disk-name</code> status argument to display the status of a particular disk.	<i>show chassis routing-engine</i>
<code>show security host-vpn security-associations</code>	Display host IPsec security association information for a specific security association or for all connections. You can configure host IPsec with the <code>host-vpn</code> statement at the [edit security] hierarchy level.	<i>show security host-vpn security-associations</i>
<code>show security host-vpn version</code>	Display the version of IPsec being used in the system.	<i>show security host-vpn version</i>
<code>show services monitoring rpm history-results</code>	Display the results stored for the specified real-time performance monitoring (RPM) probes.	<i>show services monitoring rpm history-results</i>
<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>

Table 2: New CLI Statements and Commands (Junos OS Evolved) *(Continued)*

Statement or Command	Description	Link
<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</code>	Display information about active applications on the system.	
<code>show system errors</code>	<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>

Table 2: New CLI Statements and Commands (Junos OS Evolved) *(Continued)*

Statement or Command	Description	Link
<code>show system errors history</code>	<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>
<code>show system software add-restart</code>	Display all console messages from the last in-service software upgrade (ISSU).	<i>show system software</i>
<code>show system software list</code>	Display the installed versions on the Routing Engines in the system.	<i>show system software list</i>
<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 hash-key</code>	Junos OS Evolved uses a new command to display 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 hash-key</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 3 on page 17](#).

NOTE: For the CLI commands that produce changed output, see [Table 4 on page 24](#).

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

Statement or Command	Change in Junos OS Evolved	Link
Modified Statements		
[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 system login password]	The format option for this statement is limited to the following options: (sha256 sha512).	<i>password</i>
Modified Commands		
clear ipv6 neighbors	In Junos OS Evolved, issuing the clear ipv6 neighbors command clears the cache for IPv6 neighbors in a reachable state.	<i>clear ipv6 neighbors</i>
monitor traffic interface	Starting in Junos OS Evolved 20.4R1, the write-file option at the monitor traffic interface hierarchy level takes precedence over the extensive option when you configure them 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>

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

Statement or Command	Change in Junos OS Evolved	Link
ping	<p>The following options of the ping command are deprecated:</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	<p>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.</p>	<i>request chassis routing-engine master</i>

Table 3: 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 options of the request system software add command are deprecated:</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_choice 	<i>request system software add</i>

Table 3: 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-host • validate-on-routing-engine 	
request system software delete	<p>The following options of the request system software delete command are deprecated:</p> <ul style="list-style-type: none"> • chassis • lcc • member • re-choice • scc • sfc • upgrade-group • unlink • validate • validate_choice • validate-on-host • validate-on-routing-engine 	<i>request system software delete</i>

Table 3: 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 deprecated from the request system software rollback command:</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 options of the request system software validate command are deprecated:</p> <ul style="list-style-type: none"> • chassis • lcc • member • package-options • scc • sfc 	<i>request system software validate</i>

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

Statement or Command	Change in Junos OS Evolved	Link
request system storage cleanup	<p>A new option, force-deep, is added that cleans up all user-generated files as well.</p> <p>The user is prompted to check the list of files to be deleted using the dry-run option.</p> <p>The following options are deprecated:</p> <ul style="list-style-type: none"> • re0 • re1 • routing-engine 	<i>request system storage cleanup</i>
set chassis error minor action	The offline and disable-pfe actions are not available for errors with minor severity.	<i>error</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>
show firewall	The application lsp option is introduced, which you use to display 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 connections	<p>The following options of the show system connections command are deprecated: extensive and show-routing-instance.</p> <p>The node option is introduced.</p>	<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>

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

Statement or Command	Change in Junos OS Evolved	Link
show system processes	<p>The following options of the show system processes command are deprecated:</p> <ul style="list-style-type: none"> • esc-node • health • resource-limits 	<i>show system processes</i>
telnet	<p>The following options of the telnet command are deprecated:</p> <ul style="list-style-type: none"> • bypass-routing • interface • logical-system • no-resolve • source 	<i>telnet</i>
traceroute	<p>The following options of the traceroute command are deprecated:</p> <ul style="list-style-type: none"> • logical-system • 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 4 on page 24](#).

Table 4: Changed Command Output (Junos OS Evolved)

Command	Description of Change in Output	Link
<code>clear interfaces statistics</code>	Clears not only LACP statistics but also the counters displayed in the <code>show lacp statistics interfaces</code> command.	–
<code>monitor traffic interface <i>interface-name</i></code>	When you use the command <code>monitor traffic interface <i>interface-name</i></code> 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>
<code>ping</code>	When pinging a nonresponsive route, the display output of the <code>ping</code> command does not print the number of packets sent or received or the number of packets loss.	<i>ping</i>
<code>ping <i>hostname</i> rapid</code>	When you use the <code>ping hostname rapid</code> command in Junos OS Evolved, you will see an indicator <code>. .</code> for missing replies only. No indicator <code>!!</code> for received replies is displayed.	<i>ping</i>
<code>request system snapshot</code>	Output displays the names of the directory and the individual files being copied instead of only the directory names.	<i>request system snapshot</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</i>
<code>request system software delete</code>	Output displays the version instead of the package.	<i>request system software delete</i>
<code>request system software rollback</code>	Output displays the version instead of the package.	<i>request system software rollback</i>

Table 4: Changed Command Output (Junos OS Evolved) *(Continued)*

Command	Description of Change in Output	Link
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	LACP packets on the members of an AE interface are not counted as part of the Bundle Input Statistics in the show interfaces ae <i>number</i> extensive command output.	<i>show interfaces (Aggregated Ethernet)</i>
show interfaces	Configuration of IPv6 over the re0:mgmt-* interfaces is supported.	-
show interfaces detail	Output displays the Last Flapped field with the value Never after a Routing Engine reboot. The Last Flapped field provides details of the date, time, and how long ago the interface went up. The value Never signifies that the interface never flapped.	<i>show interfaces detail</i>
show interfaces extensive	Output does not display the Packet Forwarding Engine configuration and CoS default bandwidth allocation information.	<i>show interfaces (M Series, MX Series, T Series Routers, and PTX Series Management and Internal Ethernet)</i>

Table 4: Changed Command Output (Junos OS Evolved) (Continued)

Command	Description of Change in Output	Link
<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>
<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 statistics arp</code>	After running ping on an unreachable host, output shows that counts for ARP requests received and for datagrams for an address no on the interface are incremented.	–
<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>

Table 4: Changed Command Output (Junos OS Evolved) (Continued)

Command	Description of Change in Output	Link
show system uptime	Output displays only the System booted and System-wide users information. The output does not display information on current time, system booted, protocols started, or last configured parameters. The show system uptime node command shows the other information	<i>show system uptime</i>
show task replication	Output displays the same state whether the command is run from the primary or spare Routing Engine.	<i>show task replication</i>
show version	Output of the show version command is changed to clearly show which Junos architecture is running on the device. Output of the show version node all command is revised to explicitly identify the Routing Engine in both the XML and CLI output.	<i>show version</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 5 on page 27](#). Where there is an alternative statement or command to use, it is noted in the table.

Table 5: Removed CLI Statements and Commands (Junos OS Evolved)

Statement or Command	Description
Removed Statements	

Table 5: Removed CLI Statements and Commands (Junos OS Evolved) (Continued)

Statement or Command	Description
gigether-options	The gigether-options statement at the [edit interfaces <i>interface-name</i>] hierarchy no longer appears because it is not needed. To configure link aggregation groups (LAG), use the set interfaces <i>interface-name</i> ether-options command instead.
edit system accounting traceoptions	Traceoptions for System Accounting are not supported in Junos OS Evolved.
edit system services extension-service notification	Notification service for JET applications is not supported in Junos OS Evolved.
traceoptions	The traceoptions option is removed from many of the hierarchy levels. Routing protocols (the [edit protocols] hierarchy level) is one of the applications still using traceoptions.
Removed Commands	
<ul style="list-style-type: none"> request system scripts delete request system scripts rollback 	Deprecated. AI-Scripts and Service Now are not supported on Junos OS Evolved.
request system software abort	Deprecated. There is no alternate command replacing it. The request system software add command has a built-in feature not to start an upgrade if a reboot is pending after an upgrade or rollback.
request system software (add delete) set	Deprecated. Because for Junos OS Evolved all packages are bundled into one single ISO file, the set option serves no purpose in the request system software add and request system software delete commands.
request system software in-service-upgrade	Deprecated. Use the request system software add restart command for ISSU. The request system software add command has a built-in feature not to start upgrade if a reboot is pending after an upgrade or rollback.

Table 5: Removed CLI Statements and Commands (Junos OS Evolved) (Continued)

Statement or Command	Description
<code>request system software set</code>	Deprecated. 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>	Deprecated. There are no satellite packages in Junos OS Evolved.
<code>show chassis fabric unreachability</code>	Deprecated. See the <code>show system errors</code> command for similar functionality.
<code>show chassis fabric summary</code>	The <code>show chassis fabric summary</code> command is removed. See the <code>show system errors</code> command for similar functionality.
<code>show chassis network-services</code>	Deprecated.
<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>	Deprecated. 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>	Deprecated.
<code>show interfaces em0 em1</code>	The <code>em0</code> and <code>em1</code> Ethernet management interfaces are removed. Use <code>re0:mgmt-*</code> for Routing Engine 0 (Routing Engine 1 would be <code>re1:mgmt-*</code>).
<code>show interfaces ixgbe0 ixgbe1</code>	The <code>ixgbe0</code> and <code>ixgbe1</code> internal interfaces are removed.
<code>show interfaces mac-database</code>	Deprecated. The MAC accounting and policing not supported message is displayed.

Table 5: Removed CLI Statements and Commands (Junos OS Evolved) (Continued)

Statement or Command	Description
<code>show system software detail</code>	Deprecated. 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</code> detail.
<code>show system switchover</code>	Deprecated.
<code>set system services xnm-clear-text</code>	Command is not supported and has been deprecated 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.

system storage cleanup

In Junos OS, the output of `request system storage cleanup` uses the `file` XML tag for all file types in the list of files to be deleted. In Junos OS Evolved, the output of this command groups different file types inside different XML tags.

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 (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 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**.
 - **/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.

RELATED DOCUMENTATION

[Junos OS Evolved Overview](#) | 2

Junos OS Evolved Components and Processes

IN THIS SECTION

- [Linux Kernel](#) | 36
- [Initialization Process](#) | 36

- System Epoch Management Process | 37
- System Manager Process | 37
- Management Process | 37
- Routing Protocol Process | 37
- Interface Process | 38
- Distributor Process | 38
- SNMP and MIB II Processes | 38
- Process Limits | 38

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 Evo. 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.

The clients connected to the DDS and their subscription information can be viewed using the `show platform distributor clients` command.

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.

Shell Commands for Junos OS Evolved

IN THIS SECTION

- [How to Use the Shell | 39](#)
- [Common Shell Commands | 39](#)

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 6: 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.

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.

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

2

CHAPTER

Junos OS Evolved Configuration Overview

Junos OS Evolved Configuration Basics | 43

Methods for Configuring Junos OS Evolved | 43

Junos OS Evolved Configuration from External Devices | 46

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 | 46](#)

[Methods for Configuring Junos OS Evolved | 43](#)

[Junos OS Evolved Overview | 2](#)

Methods for Configuring Junos OS Evolved

IN THIS SECTION

- [Junos OS Evolved Command-Line Interface | 44](#)
- [ASCII File | 45](#)
- [Junos XML Management Protocol Software | 45](#)
- [NETCONF XML Management Protocol Software | 45](#)
- [Configuration Commit Scripts | 45](#)

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 7: 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)	Use Junos XML protocol Perl client modules to develop custom applications for configuring information on devices that run Junos OS Evolved. Client applications use the Junos XML management protocol to request and change configuration information on supported 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)	Use NETCONF Perl client modules to develop custom applications for configuring information on devices that run Junos OS Evolved. Client applications use the NETCONF XML management protocol to request and change configuration information on 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 Extensible Stylesheet Language Transformations (XSLT) or Python.

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 Extensible Markup Language (XML) application that client applications use to request and change configuration information on supported devices. This API is customized for Junos OS Evolved, and operations in the API are equivalent to Junos OS Evolved CLI configuration mode commands. The Junos XML management protocol includes a set of Perl modules that enable client applications to communicate with a Junos XML protocol server on the router. The Perl modules are used to develop custom applications for configuring and monitoring Junos OS Evolved.

NETCONF XML Management Protocol Software

The NETCONF XML management protocol is an Extensible Markup Language (XML) application that client applications can use to request and change configuration information on supported devices. This API is customized for Junos OS Evolved, and includes features that accommodate the configuration data models of multiple vendors. The NETCONF XML management protocol includes a set of Perl modules that enable client applications to communicate with a NETCONF server on the router. The Perl modules are used 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 Extensible Stylesheet Language Transformations (XSLT) or Python.

RELATED DOCUMENTATION

[CLI Explorer](#)

[CLI User Guide](#)

[Junos OS Evolved Configuration from External Devices | 46](#)

[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 | 43](#)

[Junos OS Evolved Overview | 2](#)

3

CHAPTER

Running 3rd Party Applications with Junos OS Evolved

[Overview of Third-Party Applications on Junos OS Evolved](#) | 48

[Running Third-Party Applications in Containers](#) | 73

Overview of Third-Party Applications on Junos OS Evolved

IN THIS SECTION

- [Introduction to Third-Party Applications on Junos OS Evolved | 48](#)
- [Running Applications in Containers vs Using Signing Keys | 48](#)
- [Security Caveats | 50](#)
- [Application Pre-requisites | 50](#)
- [How to Run Applications | 51](#)
- [Using Intercept Libraries | 52](#)
- [Running Third-Party Applications in Containers | 61](#)
- [Protecting the Integrity of Junos OS Evolved with IMA | 65](#)
- [Signing Third-Party Applications to Run Natively on Junos OS Evolved | 66](#)
- [Removing Third-Party Applications | 72](#)

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 | 49](#)
- [Using Signing Keys | 49](#)

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 73](#)

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 needs 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*

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.

Application Pre-requisites

IN THIS SECTION

- [Application APIs | 50](#)

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 8 on page 51](#) provides a high-level view of the set of APIs used by applications:

Table 8: 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](#)

How to Run Applications

Applications can be launched by using Linux syntax for execution:

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


Using Intercept Libraries

IN THIS SECTION

- [Example of a Preloaded Linux Command | 53](#)
- [Interface Name Translation | 58](#)
- [Other Caveats for the Intercept Feature | 60](#)

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

Junos OS Evolved 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 9 on page 52](#):

Table 9: 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 9: 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 10 on page 59](#)) 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 10: 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 11 on page 59](#) for examples of Junos logical interface names and their Linux-compliant forms.

Table 11: 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 11: 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)
```

Other Caveats for the Intercept Feature

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

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

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

Running Third-Party Applications in Containers

IN THIS SECTION

- Deploying a Docker Container | 62
- Managing a Docker Container | 63
- Enabling Netlink or Packet IO in a Container | 63
- Selecting a VRF for a Docker Container | 64
- Modifying Resource Limits for Containers | 64

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 77](#).

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-01000000000: 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.

Protecting the Integrity of Junos OS Evolved 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 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.

SEE ALSO

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

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

IN THIS SECTION

- [Signing Keys Overview | 67](#)
- [Generating Signing Keys | 67](#)
- [Importing Signing Keys into the System Keystore and IMA Extended Keyring | 69](#)
- [Viewing the System Keystore and IMA Extended Keyring | 70](#)
- [How to Sign Applications | 71](#)
- [How to Run Signed Applications | 72](#)

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*.

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 | 68](#)
- [Generating Signing Keys Using an OpenSSL Configuration File | 68](#)

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 ]
0 = 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 ---lsrv      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!
```

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.

1. 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.
```

2. 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
```

3. 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 | 74](#)
- [Managing a Docker Container | 75](#)

- [Enabling Netlink or Packet IO in a Container | 76](#)
- [Selecting a VRF for a Docker Container | 76](#)
- [Modifying Resource Limits for Containers | 77](#)

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 77.

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-01000000000: 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.