JUNIPER NETWORKS

Junos® SDK

Applications Guide

Release
11.4

Published: 2011-10-27

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Junos® SDK Applications Guide

Revision History
November 2011—R1 Junos SDK 11.4

The information in this document is current as of the date listed in the revision history.

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 http://www.juniper.net/support/eula.html. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Abbreviated Table of Contents

Table of Contents

List of Figures

List of Tables

About This Guide

This preface provides the following guidelines for using the *Junos® SDK Applications Configuration Guide*:

Supported Routing Platforms

All applications built using the Junos SDK are supported on the Juniper Networks M Series Multiservice Edge Routers and the MX Series 3D Universal Edge Routers, with the exception that dynamic firewall filters are not supported on M40e routers. The Juniper Networks T320, T640, and T1600 Core Routers fully support these applications as well.

Among the M7i and M10i routers, the **junos_dfwd** functions (firewall filters) are supported on the following systems:

- M7i router with Enhanced Forwarding Engine
- M10i router with Enhanced Forwarding Engine
- M7i router with integrated Multiservices 100 PIC (available only with the Enhanced Forwarding Engine)

Only applications built using just the RE SDK module of the Junos SDK are supported on the Juniper Networks TX Matrix. The same is true for the Juniper Networks SRX210, SRX240, and SRX650 Services Gateways and the Juniper Networks JCS 1200 Control System.

The libdfwd statistics APIs are not supported on the Multiservices PIC. Also, the libdfwd interface creating a Level II match condition for filters is supported only on the MX Series routers.

The signing server is supported only on Linux-based and FreeBSD-based systems.

In the documentation, the term *Multiservices PIC* is equivalent to the Multiservices Dense Port Concentrator (DPC) on the MX Series routers, unless otherwise noted.

Using the Indexes

This reference contains two indexes: a complete index that includes topic entries, and an index of statements and commands only.

In the index of statements and commands, an entry refers to a statement summary section only. In the complete index, the entry for a configuration statement or command contains at least two parts:

- The primary entry refers to the statement summary section.
- The secondary entry, *usage guidelines*, refers to the section in a configuration guidelines chapter that describes how to use the statement or command.

Documentation Conventions

Table 1 on page xiv defines the text and syntax conventions used in this guide.

Table 1: Text and Syntax Conventions

Convention	Description	Examples
Bold text like this	Represents text that you type.	To enter configuration mode, type the **configure** command: user@host> **configure**
`Fixed-width text like this`	Represents output that appears on the terminal screen.	`user@host>` **show chassis alarms** `No alarms currently active`
Italic text like this	• Introduces important new terms. • Identifies book names. • Identifies RFC and Internet draft titles.	• A policy *term* is a named structure that defines match conditions and actions. • *Junos OS System Basics Configuration Guide* • RFC 1997, *BGP Communities Attribute*
Italic text like this	Represents variables (options for which you substitute a value) in commands or configuration statements.	Configure the machine's domain name: [edit] root@# **set system domain-name** *domain-name*
Text like this	Represents names of configuration statements, commands, files, and directories; interface names; configuration hierarchy levels; or labels on routing platform components.	• To configure a stub area, include the **stub** statement at the **[edit protocols ospf area area-id]** hierarchy level. • The console port is labeled **CONSOLE**.
< > (angle brackets)	Enclose optional keywords or variables.	**stub** <**default-metric** ***metric***>**;**

Table 1: Text and Syntax Conventions *(continued)*

Convention	Description	Examples
| (pipe symbol)	Indicates a choice between the mutually exclusive keywords or variables on either side of the symbol. The set of choices is often enclosed in parentheses for clarity.	**broadcast | multicast** (***string1*** **|** ***string2*** **|** ***string3***)
# (pound sign)	Indicates a comment specified on the same line as the configuration statement to which it applies.	**rsvp { # Required for dynamic MPLS only**
[ ] (square brackets)	Enclose a variable for which you can substitute one or more values.	**community name members [** ***community-ids*** **]**

Indention and braces ({ })	Identify a level in the configuration hierarchy.	[edit]
		routing-options {
		  static {
		  route default {
		  nexthop *address*;
		  retain;
		  }
		  }
		}

; (semicolon)	Identifies a leaf statement at a configuration hierarchy level.	
J-Web GUI Conventions		
Bold text like this	Represents J-Web graphical user interface (GUI) items you click or select.	• In the Logical Interfaces box, select **All Interfaces**. • To cancel the configuration, click **Cancel**.
> (bold right angle bracket)	Separates levels in a hierarchy of J-Web selections.	In the configuration editor hierarchy, select **Protocols>Ospf**.

Documentation Feedback

We encourage you to provide feedback, comments, and suggestions so that we can improve the documentation. You can send your comments to **techpubs-comments@juniper.net**, or fill out the documentation feedback form at https://www.juniper.net/cgi-bin/docbugreport/ . If you are using e-mail, be sure to include the following information with your comments:

- Document or topic name
- URL or page number
- Software release version (if applicable)

Requesting Technical Support

Technical product support is available through the Juniper Networks Technical Assistance Center (JTAC). If you are a customer with an active J-Care or JNASC support contract,

or are covered under warranty, and need post-sales technical support, you can access our tools and resources online or open a case with JTAC.

- JTAC policies—For a complete understanding of our JTAC procedures and policies, review the *JTAC User Guide* located at http://www.juniper.net/us/en/local/pdf/resource-guides/7100059-en.pdf .
- Product warranties—For product warranty information, visit http://www.juniper.net/support/warranty/ .
- JTAC hours of operation—The JTAC centers have resources available 24 hours a day, 7 days a week, 365 days a year.

Self-Help Online Tools and Resources

For quick and easy problem resolution, Juniper Networks has designed an online self-service portal called the Customer Support Center (CSC) that provides you with the following features:

- Find CSC offerings: http://www.juniper.net/customers/support/
- Search for known bugs: http://www2.juniper.net/kb/
- Find product documentation: http://www.juniper.net/techpubs/
- Find solutions and answer questions using our Knowledge Base: http://kb.juniper.net/
- Download the latest versions of software and review release notes: http://www.juniper.net/customers/csc/software/
- Search technical bulletins for relevant hardware and software notifications: https://www.juniper.net/alerts/
- Join and participate in the Juniper Networks Community Forum: http://www.juniper.net/company/communities/
- Open a case online in the CSC Case Management tool: http://www.juniper.net/cm/

To verify service entitlement by product serial number, use our Serial Number Entitlement (SNE) Tool: https://tools.juniper.net/SerialNumberEntitlementSearch/

Opening a Case with JTAC

You can open a case with JTAC on the Web or by telephone.

- Use the Case Management tool in the CSC at http://www.juniper.net/cm/ .
- Call 1-888-314-JTAC (1-888-314-5822 toll-free in the USA, Canada, and Mexico).

For international or direct-dial options in countries without toll-free numbers, see http://www.juniper.net/support/requesting-support.html .

CHAPTER 1

Software Applications on the Junos Platform

This section contains the following topics:

Introduction to Applications and the Junos SDK

The Junos Software Development Kit (SDK) allows partners of the Junos SDK program to build custom applications that run on the Juniper Networks Junos operating system (OS).

Third-party developers, here called *providers*, can use the Junos SDK to create innovative applications and manipulate existing features of the Junos OS. The term *application*, as used in this guide, refers to an application built by a provider using the Junos SDK. Documentation about the Junos SDK is aimed at one of two audiences: the providers that build the applications and the network administrators who install and configure the applications on their network devices. This guide is for network administrators.

Applications run on either a Routing Engine or a services module and so can be thought of as being either Routing Engine applications or service applications, respectively.

- Routing Engine applications run on the control plane. Typically, these applications perform network management and protocol signaling. They also initiate servers. Positioned on the control plane, Routing Engine applications can coordinate other subsystems and services. A Routing Engine is always present in any device, so these applications are always deployable without the addition of any extra hardware or software.
- Service applications run on the services plane. The services plane is specialized to enable high-performance, customized, and stateful packet processing on the transit or monitored traffic selected for servicing. Service applications may also perform operations similar to Routing Engine applications, but such activities typically supplement packet processing.

On some of the smaller Juniper Networks devices, physical modules do not necessarily plug in to a chassis. Rather a single box contains the necessary hardware. Nonetheless, applications are still supported in the control and services planes and we continue to use the Routing Engine and services modules terminology.

For more information about the Junos SDK, please contact your account team or visit the Junos SDK website at http://www.juniper.net/us/en/products-services/nos/junos/junos-sdk/.

Related Documentation

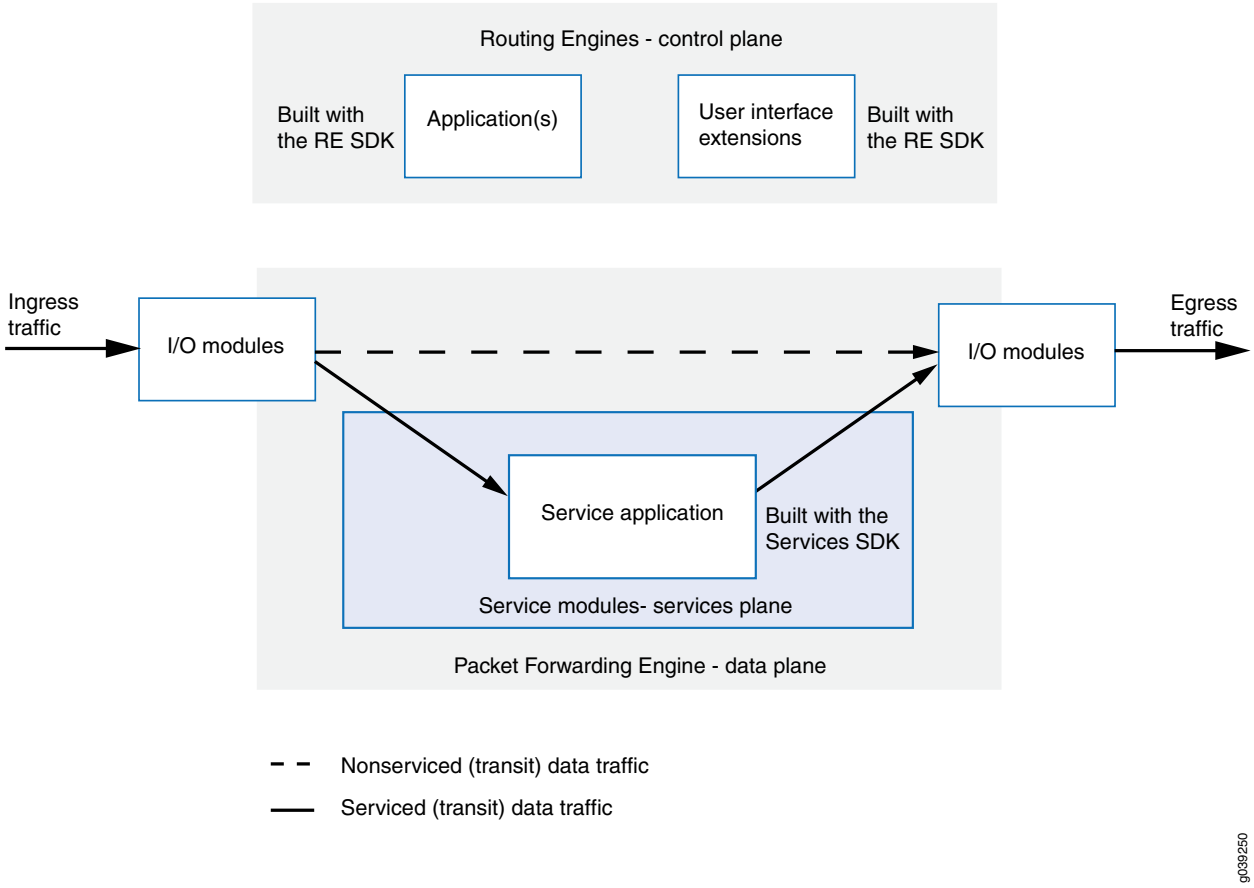
- Applications in the Junos Architecture on page 3
- Installing Application Packages on page 10

Applications in the Junos Architecture

The Junos operating system (OS) is a single network operating system that integrates routing, switching, and security. Most Juniper Networks platforms run the Junos OS and also support applications created using the Junos SDK, referred to in this guide as *applications*.

This topic presents the Junos OS architecture at a high level and in detail where it pertains to building applications. It continues explaining how the architecture fits over the hardware, and what types of applications are best suited to the environments described.

Figure 1: Architecture Summary and Traffic Paths



The Junos OS is divided into three logical elements: the control plane, the data plane, and the services plane. These main conceptual separations are summarized in Figure 1 on page 3 and described in more detail in the following sections:

- Control Plane on page 4
- Data Plane on page 4
- Services Plane on page 5
- Traffic Types on page 5

Control Plane

The control plane in the Junos OS is a logical element that manages and controls the behavior of the device and the other two planes, the data plane and the services plane. The control plane runs on the Routing Engine. The Routing Engine may be a separate physical module or may be integrated into a device. In some products the Routing Engine cards are hot-swappable physical modules in the chassis. A chassis option for a redundant Routing Engine backing up the master Routing Engine is also available. Applications can take advantage of this redundant Routing Engine to enable their own high availability.

The control plane:

- Has a global view of the device hardware and software
- Exposes the user interface and manages the native Junos OS features

The Routing Engine's key component is the Junos OS. The Junos OS is based on the FreeBSD operating system, an open-source software system. This mature, general-purpose system provides many of the essential functions of an operating system, such as scheduling resources. To transform it into a network operating system, Juniper Networks engineers have extensively modified and hardened it for the specialized requirements of networking.

On the control plane, the Junos kernel, many of the Junos daemons, and some ephemeral utility-style tools launched on demand are run. The daemons and tools that come bundled with the Junos OS are considered part of the Junos OS platform.

Applications for the control plane resemble these daemons and tools. In fact, Juniper Networks employs the Junos SDK APIs internally when building these applications, which then become part of the common platform and Junos OS software bundle. Applications developed using the Junos SDK can programmatically manipulate the states of the platform software and make use of their services in a dynamic way. The daemons that control the Junos OS user interface also allow for programmatic and seamless extensibility of the user interface. You can configure and administer a modified user interface in the same way as you do for applications.

Data Plane

The data plane in the Junos OS is a logical element that spans many aspects of a device's chassis and its modules. The data plane's role is to forward traffic according to the forwarding table, primarily formed through the routing control service on the control plane. The data plane's main extended abilities include switching, filtering, rate limiting, shaping, and other quality-of-service (QoS) functions. These functions are controlled by the control plane.

In the Junos environment and the Junos SDK architecture, the data plane abstraction is often specifically referred to as the Packet Forwarding Engine. It comprises ASIC-based hardware and software microcode to perform packet processing. Aiming to perform at fast wire speeds and within its hardware resource limits, the Packet Forwarding Engine generally defers stateful packet processing to the services plane. Applications created using the Junos SDK do not run in the data plane, which is tightly bound to the hardware.

An application running in the control or services plane can influence the packet processing mechanism in the data plane, however.

Services Plane

The services plane in the Junos OS is a logical element that can be thought of as an extension to the data plane to perform stateful services and any services non-native to the Packet Forwarding Engine. A classic example of a Junos OS service application is the stateful firewall.

In traditional forwarding devices, the data plane usually handles much of the packet processing and the services plane runs on optionally installable and hot-swappable hardware, which are generically called services modules. The services plane spans the collection of all services modules in a chassis, and a given service application can be deployed on more than one module. For the M Series, T Series, and MX Series routers, the specific modules supporting the services plane are the Multiservices PIC and the Multiservices DPC.

In security and service-oriented devices, the services plane is the primary packet processor, and the data plane merely connects a chassis containing many services modules. While these devices can perform forwarding, they are purpose built and deployed to service traffic in stateful ways implemented in software running on the services modules.

The Junos kernel on a services engine further logically divides packet handling for applications in two sub-planes. Its services plane (data plane extension) is for fast customized transit traffic processing. Its control plane is for traffic termination with the IP stack. The control plane components frequently implement a server or signaling to communicate outside the device or simply to other components on the Routing Engine.

Applications in the services sub-plane of a service module can take on two roles involving inline packet processing: transforming and monitoring. Transforming applications have access to any traffic flowing through the Packet Forwarding Engine that is selected for servicing at the given service module. They can modify, drop, hold, and create entire packets. Monitoring applications work similarly, but the packets they receive are duplicates of some traffic originally transiting the Packet Forwarding Engine. When monitoring is used, the original traffic is not impacted or serviced. The selection and configuration of either packet processing option is application specific. A hybrid application can differentiate and deal with both original and duplicated packets. Sometimes hybrid applications are named *gateways*, and may combine a server or signaling control plane component with a transforming or monitoring services plane component. Many familiar service applications, such as the Junos IPsec service, work with both styles of components.

Traffic Types

Applications deal with two different types of traffic: control traffic and data traffic.

Control traffic is in the control plane. You can classify control traffic as traffic that is either internal to the device entirely (for example, interprocess communication) or, more generally, as traffic destined to or sourced from a device address. Most addresses configured on the device belong to the master Routing Engine. For example all addresses configured on network interfaces for I/O ports and the loop back interface pertain to the

Routing Engine, so control traffic destined to those addresses is forwarded to the master Routing Engine and handled there. The Junos OS also allows the configuration of addresses on interfaces representing a service module. Traffic destined to those addresses is forwarded to and handled on the given service engine's control sub-plane.

Data traffic flows through the data plane. You can classify data traffic that relates especially to any application as the traffic selected for servicing or monitoring. This is the traffic seen in the services plane handled by transforming or monitoring applications. Data traffic naturally flows through the Packet Forwarding Engine as transit traffic, but there are application-specific mechanisms by which it can be selected for steering through the service engine's services plane. On exit, the steered traffic is re-routed and filtered in the Packet Forwarding Engine as if it was entering the device from any I/O interface.

Related Documentation

- Introduction to Applications and the Junos SDK on page 1
- Application Versions and Compatibility on page 9
- Installing Application Packages on page 10

Supported Platforms for Junos SDK

Applications may have components that execute on the control plane and on the service plane.

All control-plane-based components work across these Juniper Networks platforms:

- JCS 1200 Control System
- M Series Multiservice Edge Routers
- MX Series 3D Universal Edge Routers
- SRX210, SRX240, and SRX650 Services Gateways
- T Series Core Routers
- TX Matrix

Service-plane-based components work across these Juniper Networks routers:

- M Series Multiservice Edge Routers
- MX Series 3D Universal Edge Routers
- T Series Core Routers

All service-plane-based components work on the Multiservices 100 Physical Interface Card (PIC), the Multiservices 400 PIC, or the Multiservices Dense Port Concentrator (DPC). The Multiservices PICs are optionally installable in the M Series and T Series routers. The Multiservices DPCs are optionally installable in the MX Series routers.

NOTE: In the documentation, the term *Multiservices PIC* is equivalent to the Multiservices DPC on the MX Series routers, unless otherwise noted.

Related Documentation

- Introduction to Applications and the Junos SDK on page 1
- Applications in the Junos Architecture on page 3

CHAPTER 2

Installing and Managing Application Packages

This section includes topics on installing, enabling, managing, and removing applications:

Application Versions and Compatibility

Applications created using the Junos SDK provide binary compatibility across the different releases of the Junos OS in a particular release train. For example, all the applications developed using the Junos SDK Release 10.3 are binary compatible with Junos OS releases 10.3R1, 10.3R2, 10.3R3, and so on.

But suppose you have Junos OS Release 11.1 on your device and you use the **request system software add** command to install an application developed using Junos SDK Release 10.4. The Junos OS gracefully refuses any attempt to add the software to the device.

```
user@router> request system software add /var/tmp/jnx-example-10.4I20100917_1456.tgz
Checking compatibility with configuration
Initializing...
Using jbase-11.1B1.6
Verified manifest signed by PackageProduction_11_1_0
Verified jbase-11.1B1.6 signed by PackageProduction_11_1_0
Using /var/tmp/jnx-example-10.4I20100917_1456.tgz
Verified jnx-example-10.4I20100917_1456 signed by junipersdk-private-1
WARNING: Package 'jnx_ifinfo' is not compatible with package 'jkernel':
WARNING:     Major version number mismatch
WARNING:     (jnx_ifinfo:10 != jkernel:11)
ERROR: JUNOS version incompatible - source compatibility supported from 11.1 -
11.current_jbase_minor
```

```
ERROR: jnx-example fails requirements check
Installation failed for package '/var/tmp/jnx-example-10.4I20100917_1456.tgz'
```

NOTE: It is a general practice that applications created using the Junos SDK are built with a target Junos OS release.

Related Documentation

- Installing Application Packages on page 10
- Checking Installed Application Packages on page 14

Installing Application Packages

To install an application you install the application package or packages, and then you install the application on the router. An application package consists of binaries and their supplements. Application packages are distributed in a single **.tgz** file, which contains all the necessary resources to run the application.

To install an application on a router, you configure the statements that describe the certificate. These statements include the provider name, the type of certificate, and the scope of application deployment. The provider name identifies the provider of the application to the system and allows application packages from that provider to be installed on the router. Furthermore, each package's embedded certificate contains parameters about the provider's partnership with Juniper Networks: the license type and deployment scope parameters. As with the provider name, these parameters for the package must be allowed by the configuration in order for the package's installation to be allowed.

NOTE: If graceful routing engine switchover (GRES), or nonstop active routing (NSR) is enabled when you initiate a software installation, the software does not install properly. Make sure you issue the CLI **delete chassis redundancy** command when prompted. If GRES is enabled, it will be removed with this command. By default, NSR is disabled. If NSR is enabled, remove the **nonstop-routing** statement from the **[edit routing-options]** hierarchy level to disable it. For more information about GRES and NSR, see the *Junos OS High Availability Configuration Guide*.

To install an application:

1. Go to configuration mode of the CLI.

   ```
   user@router> configure
   Entering configuration mode
   [edit]
   user@router#
   ```

2. Go to the **[edit system extensions]** hierarchy level.

   ```
   [edit]
   user@router# edit system extensions
   ```

```
[edit system extensions]
```

3. Configure the provider name, the license type, and the deployment scope associated with the application package. This information is supplied by the application's provider.

NOTE: Older certificates have all the certificate information concatenated in the **providers** *name* statement. These older certificates are still supported, and you can install packages signed with them.

```
[edit system extensions]
user@router# set providers name license-type license deployment-scope deployment
```

For example, if **abc** is a provider name issued to a provider with a **customer** type of license and two deployment scopes, you might issue the following command. Remember, the provider name, license type, and deployment scope are specified in the application-specific documentation developed by your application's provider.

```
[edit system extensions]
user@router# set providers abc license-type customer deployment-scope [ private
  commercial ]
```

4. Make other application-specific configuration changes as required by the application's provider. This information should be in documentation developed by the provider.

5. Commit the changes to the configuration.

```
[edit system extensions]
user@router# top
[edit]
user@router# commit
commit complete
```

6. Exit the configuration mode.

```
[edit]
user@router# exit
Exiting configuration mode
```

7. If you have a rescue configuration, update the rescue configuration by issuing the **request system configuration rescue save** command again.

```
user@host> request system configuration rescue save
```

NOTE: Making sure the active configuration and the rescue configuration are consistent will prevent errors when installing applications. The system will find the same provider information in both.

8. In the operational mode of the CLI, install the application package.

```
user@router> request system software add package-name
```

You will see the application package being installed. For example, if your package's filename is **abcapps-11.1_2011.tgz**.

```
user@router> request system software add abcapps-11.1_2011.tgz
```

```
Installing package '/var/home/user/abcapps-11.1_2011.tgz'
...
Verified abcapps-11.1_2011.tgz signed by abc-enggroup
Adding abcapps...
Available space: 150060 require: 2598
Saving package file in /var/sw/pkg/abcapps-11.1_2011.tgz Saving state
for rollback ...
```

Multiple provider names can be enabled on a router. For example, if **abc** and **xyz** are provider names issued to two providers, then the following configuration enables the router for the applications built by either provider. Notice that a provider can have more than one license type and that each license type can have multiple deployment scopes:

```
[edit]
system {
  extensions {
    providers {
      abc {
        license-type customer deployment-scope [ private commercial ];
      }
      xyz {
        license-type juniper deployment-scope private;
        license-type research deployment-scope public;
      }
    }
  }
}
```

Related Documentation

- Creating and Returning to a Rescue Configuration
- Installing the Software Package on a Router with Redundant Routing Engines
- Upgrading Application Packages on page 12
- Removing Application Packages on page 13
- Checking Installed Application Packages on page 14

Upgrading Application Packages

You can upgrade to the latest version of an application by reinstalling using the new package name.

Before you reinstall an application package, you need to remove the configuration contributed by that application. See "Displaying and Deleting Configuration for Applications" on page 28 for instructions on removing application-specific configuration.

NOTE: If you do not remove the configuration before reinstalling the application package, you will get error messages. You can ignore the error messages. If you prefer not to see the error messages, remove the configuration contributed by the application before reinstalling the application package. Then you can re-add your configuration after you complete the upgrade.

To upgrade an application:

1. Delete configuration contributed by the application. See "Displaying and Deleting Configuration for Applications" on page 28.
2. Reinstall the application.

   user@router> **request system software add** ***new-sdk-package***

NOTE: If graceful routing engine switchover (GRES), or nonstop active routing (NSR) is enabled when you initiate a software installation, the software does not install properly. Make sure you issue the CLI **delete chassis redundancy** command when prompted. If GRES is enabled, it will be removed with this command. By default, NSR is disabled. If NSR is enabled, remove the **nonstop-routing** statement from the **[edit routing-options]** hierarchy level to disable it. For more information about GRES and NSR, see the *Junos OS High Availability Configuration Guide*.

TIP: After you have upgraded the software in the device and are satisfied that the new application package is successfully installed and running, take a snapshot to back up the new software to the **/altroot** and **/altconfig** file systems.

user@router> **request system snapshot**

After you run the **request system snapshot** command, you cannot return to the previous version of the software, because the running and backup copies of the software are identical.

Related Documentation

- Installing the Software Package on a Router with Redundant Routing Engines
- Installing Application Packages on page 10
- Removing Application Packages on page 13
- Checking Installed Application Packages on page 14
- Upgrading the Junos OS When Applications Are Installed on page 15

Removing Application Packages

Before you reinstall an application package, you need to remove the configuration contributed by that application. See "Displaying and Deleting Configuration for Applications" on page 28 for instructions on removing application-specific configuration.

NOTE: If you do not remove the configuration before deleting the application package, you will get error messages. You will need to do a commit to fix the configuration database. If you prefer not to see the error messages, remove the configuration contributed by the application before removing the application package.

To remove an application package:

- Delete the application package.

  user@router>**request system software delete** ***package-name***

Related Documentation

- Installing Application Packages on page 10
- Checking Installed Application Packages on page 14
- Upgrading Application Packages on page 12
- Upgrading the Junos OS When Applications Are Installed on page 15

Checking Installed Application Packages

You can get a list of the installed applications, both those created using the Junos SDK and other packages, using the **show version** command.

The following is sample output from the **show version** command:

```
user@router> show version
Hostname: router1
Model: m10i
JUNOS Base OS boot [I20070611_2103]
JUNOS Base OS Software Suite [8.5I20070611_2103]
JUNOS Kernel Software Suite [8.5I20070611_2103]
JUNOS Crypto Software Suite [8.5I20070611_2103]
JUNOS Packet Forwarding Engine Support (M/T Common) [8.5I20070611_2103]
JUNOS Packet Forwarding Engine Support (M7i/M10i) [8.5I20070611_2103]
JUNOS Online Documentation [8.5I20070611_2103]
JUNOS Routing Software Suite [8.5I20070611_2103]
JUNOS SDK Gateway Example Control Component [8.5I20070612_1932]
JUNOS SDK Gateway Example Dataplane Component [8.5I20070612_1932]
JUNOS SDK Gateway Example Management Component [8.5I20070612_1932]
```

The packages at the top of the list in the output are application packages provided by Juniper Networks with the Junos OS. The bottom of the list contains applications created using the Junos SDK. If the application package names do not start with the Junos name, then you can take them to be third-party software. However, Juniper Networks reserves the right to deploy application packages separately under the Junos name.

Related Documentation

- Installing Application Packages on page 10
- Removing Application Packages on page 13
- Upgrading Application Packages on page 12

Upgrading the Junos OS When Applications Are Installed

When you upgrade the Junos OS release, you will also need to reinstall the applications.

Before you delete an application package, you need to remove the configuration contributed by that application. See "Displaying and Deleting Configuration for Applications" on page 28 for instructions on removing application-specific configuration.

NOTE: If you do not remove the configuration before deleting the application package, you will get error messages. You can ignore the error messages. If you prefer not to see the error messages, remove the configuration contributed by the application before removing the application package. Then you can re-add your configuration after you complete the upgrade.

To upgrade the Junos OS when you have applications on your system:

1. Delete configuration contributed by the applications. See "Displaying and Deleting Configuration for Applications" on page 28.
2. Delete the application packages by following the steps described in "Removing Application Packages" on page 13.
3. Install the new version of the Junos OS. For details, see the *Junos OS Installation and Upgrade Guide*.
4. Reinstall the application created using the Junos SDK on the upgraded or downgraded version by following steps as described in "Installing Application Packages" on page 10.

NOTE: If graceful routing engine switchover (GRES), or nonstop active routing (NSR) is enabled when you initiate a software installation, the software does not install properly. Make sure you issue the CLI **delete chassis redundancy** command when prompted. If GRES is enabled, it will be removed with this command. By default, NSR is disabled. If NSR is enabled, remove the **nonstop-routing** statement from the **[edit routing-options]** hierarchy level to disable it. For more information about GRES and NSR, see the *Junos OS High Availability Configuration Guide*.

Related Documentation

- Installing the Software Package on a Router with Redundant Routing Engines
- Checking Installed Application Packages on page 14

Deploying Service Applications

There are several configurations that deal with deploying service applications. Most of these configurations are under the **[edit chassis fpc *slot-number* pic *pic-number* adaptive-services service-package extension-provider]** hierarchy level, shown here:

```
[edit chassis fpc slot-number pic pic-number adaptive-services service-package]
extension-provider {
  control-cores control-number;
  data-cores data-number;
  data-flow-affinity {
    hash-key (layer-3 | layer-4);
  }
  forwarding-db-size size;
  object-cache-size value;
  package package-name;
  policy-db-size size;
  syslog {
    facility {
      severity;
      destination destination;
    }
  }
  wired-process-mem-size mem-size;
}
```

Changing any of these settings except for the **syslog** statement causes the PIC to reboot.

Another important configuration is setting up service interfaces. The following topics cover this topic and the other configurations seen in the **[edit chassis fpc *slot-number* pic *pic-number* adaptive-services service-package extension-provider]** hierarchy level. The first three are mandatory items; the others are optional.

Each of these topics contains only generic information. Check the application-specific documentation developed by your application's provider for the appropriate configuration for your application.

Configuring Service Interfaces

To be able to configure services on your PIC, you need to configure the service interfaces. You do this by creating service sets and applying them on an interface or interfaces using the **service-set** statement at the **[edit interfaces *interface-name* unit *logical-unit-number* family (inet | inet6) service (input | output)]** hierarchy level.

The following example shows configuring a service interface (**fe-0/1/0**) by applying a service set called **Firewall-Set**:

```
[edit]
interfaces {
  fe-0/1/0 {
```

```
        unit 0 {
          family inet {
            service {
              input {
                service-set Firewall-Set;
              }
              output {
                service-set Firewall-Set;
              }
            }
            address 10.1.3.2/24;
          }
        }
      }
    }
```

For information on how to create the service sets, check your application-specific documentation, or, otherwise, follow instructions in the *Junos OS Services Interfaces Configuration Guide*.

Configuring Packages on the PIC

Applications are installed on the Multiservices PIC in one or more packages.

To designate which application package to install on a given PIC, include the **package** ***package-name*** option at the **[edit chassis fpc** ***slot-number*** **pic** ***pic-number*** **adaptive-services service-package extension-provider]** hierarchy level:

```
[edit chassis fpc slot-number pic pic-number adaptive-services service-package]
extension-provider {
  package package-name;
}
```

Up to eight packages can be installed on a PIC; however, only one data package is allowed per PIC.

NOTE: When the extension-provider statement is first configured, the PIC reboots. When any package setting is added or removed, the PIC reboots.

As of Junos OS Release 9.5, a stateful firewall plug-in is provided as part of the jbundle package. To load this plug-in on the PIC, include the **package jservices-sfw** statement at the **[edit chassis fpc** ***slot-number*** **pic** ***pic-number*** **adaptive-services service-package extension-provider]** hierarchy level.

```
user@router# show chassis
fpc 0;
    pic 0;
        adaptive-services;
            service-package;
                extension-provider;
                    control-cores 1;
                    data-cores 4;
                    object-cache-size 128;
                    package jservices-sfw;
                    policy-db-size 64;
```

```
                }
            }
        }
    }
}
```

NOTE: You cannot install both a Junos service package and an application package on the same PIC. However, you can load both the jservices-sfw package and an application package on the same PIC.

Configuring Control and Data Cores

There are eight cores in a PIC. Some cores, called *control cores*, are dedicated to running control functionality for the application. Cores dedicated to processing data for the application are called *data cores*.

To be able to deploy services on the PIC, you must configure at least one control core.

To configure control and data cores, use the **control-cores** and **data-cores** statements, respectively, at the **[edit chassis fpc *slot-number* pic *pic-number* adaptive-services service-package extension-provider]** hierarchy level:

```
[edit chassis fpc slot-number pic pic-number adaptive-services service-package]
extension-provider {
  control-cores control-number;
  data-cores data-number;
}
```

Whereas you must designate at least one core as a control core, it is advisable you designate, depending on the nature of the application, a minimum of five data cores to achieve good performance. The total number of cores, both control and data cores, that you can dedicate using the **extension-provider** statement ranges from one through eight. Any cores not configured as control or data cores are treated as *user cores*.

NOTE: When the extension-provider statement is first configured, the PIC reboots. When the number of control or data cores is changed, the PIC reboots.

Configuring Packet Distribution Settings

As of Junos OS Release 9.5, the Services SDK (a module of the Junos SDK) supports flow affinity behavior for the data CPUs. Flow affinity distribution is based on a hash distribution. Flow affinity is already the default behavior for the control CPUs, but the default behavior for distributing data packets over data cores is in a round-robin fashion.

You can change the default behavior for the data cores from round-robin to flow affinity by adding the **data-flow-affinity** statement at the **[edit chassis fpc *slot-number* pic *pic-number* adaptive-services service-package extension-provider]** hierarchy level.

```
extension-provider {
  data-flow-affinity {
```

```
    hash-key (layer-3 | layer-4);
  }
}
```

Some applications may need you to set the **hash-key** statement (consult application-specific documentation). The options for the **hash-key** statement are 3-tuple hashing (source IP, destination IP address, and IP protocol) or 5-tuple hashing (3-tuple plus source and destination TCP or UDP ports). If the **hash-key** statement is not configured, the default value is 5-tuple. There is no need to differentiate the hashing between control and data traffic.

NOTE: When the **extension-provider** statement is first configured, the PIC reboots. Either adding or removing the **data-flow-affinity** statement causes the PIC to reboot.

Setting the Forwarding Database and Policy Database Sizes

The forwarding database provides access to route information such as fast routing look-ups. It will typically have information related to route entries, their associated outgoing interfaces, and autonomous system information. The policy database stores the policies of plug-ins only. Both the forwarding database and the policy database are carved out of object cache. So the sum of both databases must be less than the total object cache.

You control the sizes of the forwarding database and the policy database using the **forwarding-db-size** and the **policy-db-size** statements, respectively, at the **[edit chassis fpc *slot-number* pic *pic-number* adaptive-services service-package extension-provider]** hierarchy level. Both statements are configured in megabytes (MB). When setting these sizes, be guided by your provider's application-specific documentation.

NOTE: Changing either the **forwarding-db-size** or the **policy-db-size** statement causes the PIC to reboot.

NOTE: You must enable the **sampling** statement at the **[edit forwarding-options]** hierarchy level for the forwarding database to be created.

Configuring Memory Settings

Configuring memory can help tune application performance. Two types of memory are provided: object cache and wired process memory. If wired process memory is exhausted, the processes use unwired memory.

```
[edit chassis fpc slot-number pic pic-number adaptive-services service-package]
extension-provider {
  object-cache-size value;
  wired-process-mem-size mem-size;
}
```

See the following topics for more information on these memory types:

- Object Cache on page 20
- Wired Process Memory on page 20

Object Cache

Object cache is shared memory. Both the size of the forwarding database and the size of the policy database are taken from object cache. Therefore, the size of object cache must be more than the sum of the sizes of both of these databases.

To configure object cache, specify a value that is a multiple of 128 MB and up to 512 MB for the Multiservices 100 PIC or up to 1280 MB for the Multiservices 400 PIC. However, if you set wired process memory as well, the maximum value for the object cache on the Multiservices 100 PIC is 128 MB and 768 MB on the Multiservices 400 PIC.

The current recommendations when configuring memory settings Multiservices PICs are as follows:

- Do not exceed a policy database size of 64 MB.
- Stay with one rule per term.
- Keep the object cache size high (1280 MB on Multiservices 400 PICs and DPCs and 512 MB on Multiservices 100 PICs).
- Do not configure anything for forwarding database.
- Keep the number of service sets per Multiservices PIC below 1000.

NOTE: When the extension-provider statement is first configured, the PIC reboots. Changing the object cache size, the policy database size, or the forwarding database size on a running system causes the PIC to reboot.

Wired Process Memory

Wired process memory, or Big TLB, is memory used by the operating system that is pretty much "off limits" to another application. If wired process memory is exhausted, the processes use unwired memory.

The size of wired process memory is configurable. To reserve Big TLB, configure the **wired-process-mem-size** statement at the **[edit adaptive-services service-package extension-provider]** hierarchy level.

NOTE: When the wired-process-mem-size statement is first changed, the PIC reboots.

As of Junos OS Release 11.2, the number of processes supported by Big TLB is also configurable. When you configure multiple Big-TLB-supported processes, the physical memory is divided equally among the number of processes that use it. For example, if

mem-size is 1024 MB and the number of processes (***num-procs***) supported is 8, then each process is allocated 128 MB of physical memory. Thus, the amount of physical memory that is allocated to each process is ***mem-size/num-procs***.

To specify the number of processes that use Big TLB, use the **wired-max-processes** statement at the **[edit adaptive-services service-package extension-provider]** hierarchy level. The amount of memory that can be reserved for Big TLB is platform dependent.

The amount of object cache is also dependent on the amount of wired memory. The following table lists details per platform.

Table 2: Wired Memory and Object Cache Combinations in Multiservices PICs

Device	Max. Object Cache (MB)	Max. Wired Memory (MB)	Max. Number of Processes
Multiservices 100 PIC	512	0	0
	0	512	8
Multiservices 400 PIC	1280	0	0
Multiservices 500 PIC Multiservices DPC	256	1024	8

Configuring System Log Messages

To record or view system log messages on a specific PIC, include the **syslog** statement at the **[edit chassis fpc *slot-number* pic *pic-number* adaptive-services service-package extension-provider]** hierarchy level:

```
[edit chassis fpc slot-number pic pic-number adaptive-services service-package]
extension-provider {
  syslog {
    facility {
      severity;
      destination destination;
    }
  }
}
```

NOTE: When the extension-provider statement is first configured, the PIC reboots. After that, changing system log settings does not cause the PIC to reboot.

Each system log message belongs to a *facility*, which is a group of messages that are either generated by the same software process or concern a similar condition or activity. Each message is also preassigned a *severity level*, which indicates how seriously the triggering event affects router functions.

For the Junos SDK, there are four values for the ***facility*** statement that log either actions performed or errors encountered by the following entities:

- **daemon**—Various system processes.
- **external**—Local external applications.
- **kernel**—The PIC kernel.
- **pfe**—The Packet Forwarding Engine.

The ***severity*** statement has the same values as it does in the native Junos operating system (OS). See Table 3 on page 22 for possible values.

Table 3: Severity Levels for SDK Syslog Messages

Level	Description
any	Include all severity levels.
none	Disable logging of the associated facility to a destination.
emergency	System panic or other condition that causes the routing platform to stop functioning.
alert	Conditions that require immediate correction, such as a corrupted system database.
critical	Critical conditions, such as hard errors.
error	Error conditions that generally have less serious consequences than errors in the emergency, alert, and critical levels.
warning	Conditions that warrant monitoring.
notice	Conditions that are not errors but might warrant special handling.
info	Events or nonerror conditions of interest.

Enhancements to the existing infrastructure make debugging on the Multiservices PIC easier by giving the user the option of redirecting the log messages to either the Routing Engine (**routing-engine**) or to the console of the PIC (**pic-console**).

The user does not have to specify a destination for the messages; by default all messages go to the **/var/log** directory on the Routing Engine. When the syslog destination is configured to redirect the log messages to the Routing Engine, you can use the **set system syslog** command, a command available in the Junos OS CLI, to override the syslog settings made on the Multiservices PIC.

To record or view system log messages on a specific PIC, include the **syslog** statement. System log information is passed to the Routing Engine and put in the **/var/log** directory.

CHAPTER 3

Working with Applications

The following topics describe how to work with applications in a general way. This information does not replace information from application-specific documentation developed by your provider.

Application Execution

Application execution includes the following topics. For more information on system processes, see the *Junos OS System Basics Configuration Guide*.

Verifying the Execution Status of System Processes

To see the execution status of system processes, use the **show system processes wide** operational command. The **wide** option is optional, but you must use it to display the provider name of your application.

show system processes wide

```
user@host> show system processes wide
     PID  TT  STAT      TIME PROVIDER COMMAND
  ...
       9  ??  DL     0:00.01          [pagedaemon]
  ...
    6463  ??  DL     0:00.18          [md9]
    6738  ??  S      0:00.44          /usr/sbin/mgd -N
    7001  ??  S      0:00.12 jnx      /opt/sbin/jnx-exampled -N
    7063  ??  Ss     0:00.03          mgd: (mgd) (regress)/dev/ttyp0 (mgd)
```

Disabling a System Process

You can disable Junos OS processes.

CAUTION: Never disable any of the software processes unless instructed to do so by a Customer Support engineer.

To disable a software process:

1. Specify the appropriate process to disable at the **[edit system]** hierarchy level using the **processes** statement.

   [edit system]
   user@router# **set processes** *process-name* **disable**

2. You can later go back and similarly enable the process.

   [edit system]
   user@router# **set processes** *process-name* **enable**

TIP: The *process-name* variable is one of the valid process names. You can obtain a complete list of process names by using the CLI command completion feature.

Configuring Application Failure Settings

For routers or switches with redundant Routing Engines, you can configure the router or switch to switch to backup media that contains a version of the system if a software process fails repeatedly. You can configure the router or switch to fail over to either backup media or the other Routing Engine. When you configure a failover option, the router will try to reboot a few times (within one minute) before going to the failover option you specify.

To configure automatic switchover to backup media if a software process fails:

- Include the **failover** statement at the **[edit system processes** *process-name***]** hierarchy level:

  [edit system]
  user@router# **set** *process-name* **failover (alternate-media | other-routing-engine)**

TIP: The *process-name* variable is one of the valid process names. You can obtain a complete list of process names by using the CLI command completion feature.

Resource Limits Imposed on Applications

The Junos SDK has four levels of policies to ensure that applications have the minimal impact on the native Junos operating system and system operations. These policies impose limits on the system resources the application uses. Each succeeding policy level overrides the previous level's settings, provided the constraints are within the previous level's settings.

Level I policy is the default global policy generated by Juniper Networks. Level II is a per-provider level policy that enables different resource limits per provider. A Level III policy is implemented in the policy file that providers write for each application package. The Level IV policy is set by you, the administrative user using the **resource-limits** statement at the **[edit system extensions]** hierarchy level.

NOTE: This documentation is not application-specific documentation. Features documented in this guide are generic to all applications. Refer to the application-specific documentation developed by your applications' provider for detailed instructions.

- Displaying Resource Limits on page 25
- Restricting Resource Limits on page 26

Displaying Resource Limits

To display the applied policies, use the **show system processes resource-limits process-name *process-ui-name*** operational command. The following example configuration, when committed, applies resource limits for an application package jnx-example and overrides it with process-level settings for the process jnx-example-service:

```
[edit system extensions]

user@router# show
resource-limits {
    package jnx-example {
        resources {
            memory {
                stack-size 4m;
            }
        }
    }
    process jnx-example-service {
        resources {
            file {
                size 4m;
            }
        }
    }
}
```

Using the **show system processes resource-limits process-name** command, the output for jnx-example-foo-binary, part of package jnx-example, looks like the following output because the package-level settings are applied on it:

```
user@router> show system processes resource-limits process-name jnx-example-foo-binary
Resource Limits:
  Area                       Max. allowed     Max. configurable
  memory/stack-size                  4MB                   8MB
  memory/data-size                  32MB                  32MB
  memory/resident-set-size          24MB                  24MB
  memory/locked-in                  16MB                  16MB
  cpu/priority                        10                    10
  file/open                           64                    64
```

The output for jnx-example-service looks like this:

```
user@router> show system processes resource-limits process-name jnx-example-service
Resource Limits:
  Area                       Max. allowed     Max. configurable
  file/size                          4MB             unlimited
  file/open                           64                    64
  cpu/priority                        10                    10
  memory/stack-size                  8MB                   8MB
  memory/data-size                  32MB                  32MB
  memory/resident-set-size          24MB                  24MB
  memory/locked-in                  16MB                  16MB
```

For more detail on the **show system processes resource-limits process-name** ***process-ui-name*** operational command, see its command summary.

Restricting Resource Limits

The Level IV policy is set by the administrative user using the **resource-limits** statement at the **[edit system extensions]** hierarchy level.

The limits imposed by a Level IV policy can be configured either by package or by individual processes in the package. Limits defined for individual processes override the limits defined for an entire package. Any limits not set as Level IV limits inherit the limits from Level III if they exist or from Level II.

The following hierarchy shows all the statements for setting resource limits:

```
[edit system extensions]
resource-limits {
  package package-name {
    resources {
      cpu {
        priority number;
        time seconds;
      }
      file {
        core-size bytes;
        open number;
        size bytes;
      }
      memory {
        data-size bytes;
        locked-in bytes;
```

```
            resident-set-size bytes;
            socket-buffers bytes;
            stack-size bytes;
          }
        }
      }
      process process-ui-name {
        resources {
          cpu {
            priority number;
            time seconds;
          }
          file {
            core-size bytes;
            open number;
            size bytes;
          }
          memory {
            data-size bytes;
            locked-in bytes;
            resident-set-size bytes;
            socket-buffers bytes;
            stack-size bytes;
          }
        }
      }
    }
```

If an application exceeds any of the imposed limits, the router logs it. For example, if a process tries to exceed its stack size, the process is terminated and the system generates a core file.

Level IV policies can be more restrictive than previous policy levels, but they cannot ease the limits set by the other levels. If you try to commit a resource limit that is higher (less stringent) than the inherited value, the commit operation fails with the following error message:

```
[edit system extensions resource-limits]
  'process jnx-example-service'
    Limit validation failed for program 'jnx-example-service', resource 'file'
limit 'open': raising limits defined in role 'Provider_Daemon' is not allowed.
commit complete

[edit system extensions resource-limits]
```

Level IV policies can be applied either during runtime of the application or before the application gets started. However, if the application was already running when a new limit is applied, the application must be restarted manually in order to allow for the new limits to take effect.

If you delete a resource configuration, the setting goes back to the limits from the assigned role in the manifest file (Level II or Level III).

Application User Interface Extensions

With the Junos SDK, providers can extend the Junos OS user interface. The Junos OS has two configuration mode commands developed using the Junos SDK that help you work with applications. The following topics describe these commands in detail:

Displaying the Detail for Application Configurations

The **show | display detail** command is not new but has been updated to show application package information. The complete form of the command is as follows:

show <*statement-path*> | display detail

The **show | display detail** command displays the characteristics, descriptions, and constraints of each configuration statement in the Junos operating system (OS) configuration schema using comment lines. The configuration schema is piped through to the **display detail** command. The **show** command displays the entire user-defined configuration unless it is limited to a particular branch of the configuration scheme by the optional ***statement-path*** variable.

Generally, the information displayed is help strings and the permission bits required to add or modify the configuration statement. For configuration statements that are defined by an application package, the name of the package that defines the statement is also displayed. If a Junos statement is redefined by an application package, the package name is listed. However, if a configuration statement is defined by the native Junos OS only, no package name is displayed.

Displaying and Deleting Configuration for Applications

To display or delete the configuration for a specific application package, use the **extension *package-name* (show | delete)** configuration mode command:

extension *package-name* (show | delete) <*section*>

This command filters for configuration based on the package named in the command, starting at the top of the configuration hierarchy, or, if you use the ***section*** option, starting at the hierarchy level indicated by the ***section*** option. The output displays the configuration of all matches.

NOTE: Remove all the soon-to-be invalid configurations before removing the application package. See "Removing Application Packages" on page 13.

To delete application configuration:

1. Display any configuration specific to the application.

   ```
   [edit]
   user@router# extension package-name show
   ```

2. When you are satisfied that this is the configuration you want to delete, issue the **extension *package-name* delete** command. Use the ***section*** option to limit the scope of the deletion if needed.

   ```
   [edit]
   user@router# extension package-name delete
   ```

NOTE:

Matches for the extension show Command

Matches for the **extension *package-name* show** command include those packages whose package names exactly match the value of ***package-name*** as well as those whose names have the same root but may have longer names, similar to a wildcard situation.

The following example shows simplified output illustrating how the **extension *package-name* show** command works. Suppose a router has packageA, packageB, and packageAB installed and you issue the command **extension packageA show**. The output displays configuration for packages with names that not only exactly match packageA but also have roots that match (in this case, packageAB):

```
user@router# extension packageA show
system {
   packageA {
      ....
   }
   packageAB {
      ....
   }
}
```

Matches for the extension delete Command

In contrast, the **extension** ***package-name*** **delete** command treats the value of ***package-name*** as a literal. It deletes only configuration contributed by the package whose package name exactly matches the given value.

NOTE: A configuration schema defined in any native Junos package is never deleted by the removal of any third-party package.

For more examples of output for the **extension** ***package-name*** **(show | delete)** command, see "Example: Displaying and Deleting Configuration Contributed by Applications" on page 37.

Application Health Monitoring

Using the Junos SDK, providers can have their applications request and manage system resources. Some of this resource utilization is persistent across, for example, reboots or the restart of the application. Some system tasks such as deleting a package, disabling and application, or accessing shared resources require that resources be cleaned up by entities other than the application itself. Resources that are known to need cleaning up include the following:

- GENCFG blobs
- SYSV shared memory segments
- SYSV semaphores
- Temporary files

Currently, the **traceoptions** statement is the only CLI statement available for configuring resource cleanup.

To configure tracing operations for resource cleanup operations, include the **traceoptions flag** option for selectively turning the debugging of trace messages on or off:

```
[edit]
system {
   processes {
      resource-cleanup {
         traceoptions {
            file filename files number match regex size size (world-readable |
               no-world-readable);
            flag flag;
            level level;
            no-remote-trace;
         }
      }
   }
}
```

The available flags for the **traceoptions** statement include:

- **all**—Enable all trace option flags.
- **events**—Display process state change and cleanup events.
- **gencfg**—Display GENCFG blobs recorded for cleanup.
- **sysvsem**—Display SYSV semaphores recorded for cleanup.
- **sysvshm**—Display SYSV shared memory segments recorded for cleanup.
- **ui**—Display tracing messages for UI operational commands.

CHAPTER 4

Traffic Configuration with Applications

This section covers:

Configuring Traffic Sampling for Junos SDK Applications

You enable sampling for Junos SDK-built applications much as you do for with the regular Junos operating system (OS). But only family inet and family inet6 are supported for Junos SDK-built applications receiving traffic on an **ms-** interface.

Developed for Junos SDK-built applications, the **extension-service** statement provides a section of the configuration hierarchy in which the provider of your application may have added its own traffic-monitoring configuration statements. The **extension-service** statement is available at the following hierarchy levels:

- **[edit services service-set *service-set-name*]**
- **[edit forwarding-options sampling family *family* output]**
- **[edit forwarding-options sampling instance *instance-name* family *family* output]**

To enable sampling for applications and be able to use any application-specific configuration statements your application provider may have added, you must include the **extension-service** statement at the **[edit forwarding-options sampling family *family* output]** or **[edit forwarding-options sampling instance *instance-name* family *family* output]** hierarchy level. For application-specific configuration guidelines, see the documentation provided with your application.

NOTE: If you use the **extension-service** statement, the only other statement you can include at the **[edit forwarding-options sampling family *family* output]** hierarchy level is the **interface** statement. In this case, you must set the **interface** statement to an interface with an **ms-** prefix.

[edit]

```
forwarding-options {
  sampling {
    family family {
      output {
        extension-service service-name {
          provider-specific rules;
        }
        interface ms-fpc/pic/port;
      }
    }
  }
}
```

Related Documentation

- Example: Traffic Sampling on a Multiservices PIC on page 41

Configuring Service Sets for Junos SDK Applications

A *service set* is a collection of policies taken from multiple services that can be applied as a unit to traffic coming to the PIC.

For SDK applications, to include a defined service in a service set, you must reference it using the **extension-service** statement at the **[edit services service-set *service-set-name*]** hierarchy level.

```
[edit]
services {
  service-set service-set-name {
    extension-service service-name1;
    extension-service service-name2;
  }
}
```

Up to two SDK plug-ins are supported per PIC and can be chained together in one service set.

NOTE: If the **extension-service** statement is used, the **service-order** statement is mandatory.

To specify the order of the policies within a service set, configure the **service-order** statement at the **[edit services extension-service]** hierarchy level.

Related Documentation

- Configuring the Service Order for Junos SDK Service Sets on page 35
- Example: Junos SDK Service Set Configuration on page 40

Configuring the Service Order for Junos SDK Service Sets

The service order is the order in which services are applied for a given service set.

To configure the service order, include the **service-order** statement at the **[edit services service-set *service-set-name* extension-service]** hierarchy level.

```
[edit]
services {
   service-set service-set-name {
      extension-service service-name1;
      extension-service service-name2;
      service-order {
         forward-flow [ service-name1 service-name2 ];
         reverse-flow [ service-name1 service-name2 ];
      }
   }
}
```

NOTE: If the extension-service statement is specified, the service-order statement is mandatory. Service order should not be configured for native Junos internal services. For the internal services, there is a default service order.

The **service-order** statement must include all services defined in the service set. It is mandatory to specify the forward-flow service order and the reverse-flow service order. If the reverse-flow service order is not specified, the reverse-flow order is the reverse of the forward-flow service order. The exception to this is for the sampling service set type. If a service set is a sampling service set and the reverse-flow service order is not configured, all sampled traffic is considered to be forward traffic.

To change the service order, delete the service order elements and then add them again in the new order.

Related Documentation

- Configuring Service Sets for Junos SDK Applications on page 34
- Configuring the Sampling Service Set on page 35

Configuring the Sampling Service Set

In addition to next-hop and interface service sets there is also a *sampling service set*. Interface and next-hop service sets are explained in the *Junos OS Services Interfaces Configuration Guide*.

The sampling service set is configured using the **sampling-service** statement at the **[edit services service-set *service-set-name*]** hierarchy level.

```
[edit]
services {
   service-set service-set-name {
```

```
    sampling-service {
      service-interface interface-name;
    }
  }
}
```

The *service interface* is the interface the sampling is taken from. In the case of a sampling service set, the service interface must be a Multiservices PIC interface with a subunit number of 0 (zero). The subunit number defaults to 0. The **reverse-flow** statement is not mandatory. All sampled traffic is considered to be forward traffic. If you set the **reverse-flow** statement, it is ignored.

The next example makes sure that any sampled packet to the **ms-6/1/0.0** interface has the service plug-ins **plugin1** and **plugin2** applied to it.

```
[edit]
services {
  service-set sset1 {
    sampling-service {
      service-interface ms-6/1/0;
    }
    service-order {
      forward-flow [ plugin1 plugin2 ];
    }
  }
}
```

CHAPTER 5

Examples

This section contains the examples for working with applications built using the Junos SDK:

Example: Displaying and Deleting Configuration Contributed by Applications

In the following example, only the **sdk-backup-server** statement at the **[edit system radius-server 10.1.1.1]** hierarchy level and the **sdk-proto1** statement at the **[edit system protocols]** hierarchy level are contributed by the application package **sdk-pkg1**.

Output from show Command

If you use the **show** command at the top of the hierarchy, all configuration is shown.

```
[edit]
user@router# show
system {
   radius-server {
      10.1.1.1 {
         timeout 10;
         sdk-backup-server 10.1.1.2; # Contributed by sdk-pkg1
      }
   }
}
protocols {
   ospf {
      ....
   }
   sdk-proto1 { # Contributed by sdk-pkg1
      ....
   }
}
```

Output from extension show Command

In contrast, the following is the output from the **extension sdk-pkg1 show** command. Notice that this command selects configurations contributed by and leading to the package named in the command. The **extension sdk-pkg1 show** command filters out the **timeout** and **ospf** commands from the displayed output, which were not contributed by the **sdk-pkg1** package.

```
[edit]
user@router# extension sdk-pkg1 show
system {
   radius-server {
      10.1.1.1 {
         sdk-backup-server 10.1.1.2;
      }
   }
}
protocols {
   sdk-proto1 {
      ....
   }
}
```

Output After the extension delete Command

Continuing with the same example, notice how the configuration changes when the **extension sdk-pkg1 delete** command is used. Using this command, you can delete all user-defined configuration statements related to the sdk-pkg1 package.

TIP:

This one command accomplishes the same thing as issuing the following two commands:

- **delete system radius-server 10.1.1.1 sdk-backup-server**
- **delete protocols sdk-proto1**

```
[edit]
user@router# extension sdk-pkg1 delete
[edit]
user@router# show
system {
   radius-server {
      10.1.1.1 {
         timeout 10;
      }
   }
}
protocols {
   ospf {
      ....
   }
}
```

Related Documentation

- Displaying and Deleting Configuration for Applications on page 28

Example: Configuring Separate Resource Limits for a Process

In this example there are 10 processes in the application package, and the administrative user wants to limit the number of open files and the stack size but does not want to apply the same limits to the jnx-example-service process. To do this, the user must specify limits for the jnx-example package and for the jnx-example-service process. Any limits not set inherit the Level III limits (if any) or Level II limits (if no Level III limits exist).

```
user@router# show
resource-limits {
   package jnx-example {
      resources {
         file {
            open 8;
         }
         memory {
            stack-size 4m;
         }
      }
   }
   process jnx-example-service {
      resources {
         memory {
            resident-set-size 8m;
         }
      }
   }
}

[edit]

user@router# commit
```

The **show system processes resource-limits process-name** command output shows a maximum configurable column, which, in the following examples, has been omitted to make room for showing from which policy level the maximum allowed values come from.

```
user@router> show system processes resource-limits process-name jnx-foo-service
Resource Limits
   Area                     Max. allowed
   cpu/priority                       10  <-- inherited from Level II
   file/open                           8  <-- defined in config
   memory/data-size                 32MB  <-- inherited from Level II
   memory/locked-in                 16MB  <-- inherited from Level II
   memory/resident-set-size         24MB  <-- inherited from Level II
   memory/stack-size                 4MB  <-- defined in config
```

The above output is the same for all other programs in the jnx-example package except the jnx-example-service process, for which there is a special configuration. The process-level configuration overrides the package-level configuration.

```
user@router> show system processes resource-limits process-name jnx-example-service
Resource Limits
   Area                     Max. allowed
   cpu/priority                       10  <-- inherited from Level II
   file/open                          64  <-- inherited from Level II
   memory/data-size                 32MB  <-- inherited from Level II
   memory/locked-in                 16MB  <-- inherited from Level II
```

```
memory/resident-set-size          8MB   <-- defined in config
memory/stack-size                 8MB   <-- inherited from Level II
```

Related Documentation

- Resource Limits Imposed on Applications on page 25
- Restricting Resource Limits on page 26
- Displaying Resource Limits on page 25

Example: Junos SDK Service Set Configuration

In the following configuration example, the **acme-svc1** service is defined by three rules (their content is unspecified) and the **acme-svc2** service is defined by a rule set made up of an unspecified number of rules. In this example, these services are defined at the **[edit acme services]** hierarchy level.

```
[edit]
acme {
  services {
    acme-svc1 { #Provider-defined service
      svc1-rule1 { # First rule's name
        . . . # First rule defined
      }
      svc1-rule2 { # Second rule's name
        . . . # Second rule defined
      }
      svc1-rule3 { # Third rule's name
        . . . # Third rule defined
      }
    }
    acme-svc2 { # Provider-defined service
      rule-set svc2-rule-set { # Rule-set name
        [ rules rule-names ]; # Rules definitions start here
      }
    }
  }
}
```

At the **[edit services]** hierarchy level (no intervening "acme" level here), the **service-set sset1** is defined by referencing the three rule names for **acme-svc1** and the one rule set name for **acme-svc2** using the **service-set *service-set-name* extension-service** statement at the **[edit services service-set *service-set-name*]** hierarchy level.

The service order is also configured at the **[edit services service-set *service-set-name*]** hierarchy level.

The following is an example of configuring service sets, extension service rules, and the service order:

```
[edit]
services {
  service-set sset1 {
    extension-service acme-svc1 {
      svc1-rule1;
```

```
        svc1-rule2;
        svc1-rule3;
      }
      extension-service acme-svc2 {
        rule-set svc2-rule-set;
      }
      /* Now define the order */
      service-order {
        forward-flow [acme-svc1 acme-svc2];
        reverse-flow [acme-svc1 acme-svc2];
      }
    }
  }
```

Related Documentation

- Configuring the Sampling Service Set on page 35
- Configuring the Service Order for Junos SDK Service Sets on page 35
- Configuring Service Sets for Junos SDK Applications on page 34

Example: Traffic Sampling on a Multiservices PIC

The following example shows a firewall filter **sample-monitor**, which, when attached to an interface, ensures that all traffic entering that interface with a source address matching address **10.1.1.1** is sampled and sent to the output interface **ms-2/0/0**.

```
[edit]
firewall {
  family inet {
    filter sample-monitor {
      term sample-term {
        from {
          source-address {
            10.1.1.1/32;
          }
        }
        then {
          sample;
          accept;
        }
      }
    }
  }
}
forwarding-options {
  sampling {
    input {
      family inet {
        rate 1;
      }
    }
    output {
      extension-service abc-sample {
        provider-specific rules;
      }
```

```
        interface ms-2/0/0;
    }
}
```

You can attach the firewall filter created in the above sample to any interface in your network.

Related Documentation

- Configuring Traffic Sampling for Junos SDK Applications on page 33

CHAPTER 6

Summary of Application Configuration Statements

This chapter provides a reference for each of the configuration statements that relate to configuring applications built by the Junos SDK. The statements are organized alphabetically.

control-cores

Syntax control-cores *control-number*;

Hierarchy Level [edit chassis fpc *slot-number* pic *pic-number* adaptive-services service-package extension-provider]

Release Information Statement introduced in Junos OS Release 9.0.

Description Configure control cores. Any cores not configured as either control or data cores are treated as user cores. When the number of control cores is changed, the PIC reboots.

Options ***control-number***—Number of control cores. At least one core must be a control core.
Range: 1 through 8

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

Related Documentation
- Configuring Control and Data Cores on page 18
- data-cores on page 44

data-cores

Syntax data-cores *data-number*;

Hierarchy Level [edit chassis fpc *slot-number* pic *pic-number* adaptive-services service-package extension-provider]

Release Information Statement introduced in Junos OS Release 9.0.

Description Configure data cores. Any cores not configured as either data or control cores are treated as user cores. When the number of data cores is changed, the PIC reboots.

Options ***data-number***—Number of data cores. Although it is not mandatory to dedicate any cores as data cores, it is advisable, depending on the nature of the application, to dedicate a minimum of five as data cores to achieve good performance.

Range: 0 through 7

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

Related Documentation
- Configuring Control and Data Cores on page 18
- control-cores on page 43

data-flow-affinity

Syntax
```
data-flow-affinity {
  hash-key (layer-3 | layer-4);
}
```

Hierarchy Level [edit chassis fpc *slot-number* pic *pic-number* adaptive-services service-package extension-provider]

Release Information Statement introduced in Junos OS Release 9.5.

Description Enable flow affinity distribution for packets over data CPUs on the PIC. Once enabled, the default behavior distributing data packets changes from a round-robin distribution to a flow affinity distribution based on a hash distribution. Adding or deleting this statement causes the PIC to reboot.

The statements are explained separately.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

Related Documentation
- Configuring Packet Distribution Settings on page 18

destination

Syntax destination *destination*;

Hierarchy Level [edit chassis fpc *slot-number* pic *pic-number* adaptive-services service-package extension-provider syslog *facility*]

Release Information Statement introduced in Junos OS Release 10.1.

Description Configure where log messages go. By default, all messages go to the **/var/log** directory on the Routing Engine. Enhancements to the existing infrastructure make debugging on the Multiservices PIC easier by giving the user the option of redirecting log messages. When the **syslog destination** statement is configured to redirect the log messages, you can use the **set system syslog** command, a command available in the native Junos OS CLI, to override the syslog settings made on the Multiservices PIC.

Options ***destination***—Choose one of the following options:

- **routing-engine**—Forward log messages to the Routing Engine.
- **pic-console**—Forward log messages to the console of the PIC.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

Related Documentation

- Configuring System Log Messages on page 21
- extension-provider on page 46

extension-provider

Syntax

```
extension-provider {
  control-cores control-number;
  data-cores data-number;
  data-flow-affinity {
    hash-key (layer-3 | layer-4);
  }
  forwarding-db-size size;
  object-cache-size size;
  package package-name;
  policy-db-size size;
  syslog {
    facility {
      severity;
      destination destination;
    }
  }
  wired-process-mem-size mem-size;
}
```

Hierarchy Level [edit chassis fpc *slot-number* pic *pic-number* adaptive-services service-package]

Release Information Statement introduced in Junos OS Release 9.0.

Description Configure an application on a PIC. When the **extension-provider** statement is first configured, the PIC reboots.

The statements are explained separately.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

Related Documentation

- Configuring Control and Data Cores on page 18
- Configuring Packet Distribution Settings on page 18
- Configuring Memory Settings on page 19
- Configuring Packages on the PIC on page 17
- Configuring System Log Messages on page 21

extension-service

Syntax

```
extension-service service-name {
  provider-specific rules;
}
```

Hierarchy Level

[edit forwarding-options sampling instance *instance-name* family (inet |inet6) output]
[edit forwarding-options sampling family (inet |inet6) output]
[edit services service-set *service-set-name*]

Release Information

Statement introduced in Junos OS Release 9.0.

Description

Define a customer specific sampling configuration.

Define a service set or traffic monitoring for applications using application-specific configuration guidelines.

NOTE: If the extension-service statement is specified while configuring a service set, the service-order statement is mandatory.

Options

provider-specific rules—Provider-specific subhierarchy for services and service sets. See the application-specific documentation for details.

service-name—Name of the service.

Required Privilege Level

system—To view this statement in the configuration.
system-control—To add this statement to the configuration.

Related Documentation

- service-order on page 61
- sampling

extensions

Syntax

```
extensions {
  providers {
    provider-id {
      license-type license deployment-scope [ deployments ];
    }
  }
  resource-limits {
    package package-name {
      resources {
        cpu {
          priority number;
          time seconds;
        }
        file {
          core-size bytes;
          open number;
          size bytes;
        }
        memory {
          data-size bytes;
          locked-in bytes;
          resident-set-size bytes;
          socket-buffers bytes;
          stack-size bytes;
        }
      }
    }
    process process-ui-name {
      resources {
        cpu {
          priority number;
          time seconds;
        }
        file {
          core-size bytes;
          open number;
          size bytes;
        }
        memory {
          data-size bytes;
          locked-in bytes;
          resident-set-size bytes;
          socket-buffers bytes;
          stack-size bytes;
        }
      }
    }
  }
}
```

Hierarchy Level [edit system]

Release Information Statement introduced in Junos OS Release 9.0.

Description Configure extensions to Junos operating system (OS). You must turn configure the **providers *provider-id*** statement to enable application packages to be deployed and run on the router.

The statements are explained separately.

Required Privilege Level admin—To view this statement in the configuration.
admin-control—To add this statement to the configuration.

Related Documentation
- Deploying Applications on the Router

forwarding-db-size

Syntax forwarding-db-size *size*;

Hierarchy Level [edit chassis fpc *slot-number* pic *pic-number* adaptive-services service-package extension-provider]

Release Information Statement introduced in Junos OS Release 9.2.

Description Configure the size of the forwarding database (FDB). When this setting is changed, the PIC reboots.

NOTE: You need to enable the **forwarding-options sampling** statement for the FDB to be created.

Options ***size***—Size of the FDB, in megabytes (MB). The size of the FDB and the size of the policy database together must be smaller than the size of the object cache.

Range: 0 through 12879 MB

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

Related Documentation
- Configuring Memory Settings on page 19
- policy-db-size on page 53
- wired-process-mem-size on page 67
- object-cache-size on page 51

hash-key

Syntax hash-key (layer-3 | layer-4);

Hierarchy Level [edit chassis fpc *slot-number* pic *pic-number* adaptive-services service-package extension-provider data-flow-affinity]

Release Information Statement introduced in Junos OS Release 10.2.

Description Set the hashing distribution of flow affinity. This is an optional setting. Once the **data-flow-affinity** statement is enabled, you may need to choose the hashing distribution. Modifying this statement causes the PIC to reboot.

Default If you do not configure the **hash-key** statement, the hashing distribution is 5-tuple hashing, or **layer-4**.

Options **layer-3**—3-tuple hashing (source IP address, destination IP address, and IP protocol).

layer-4—5-tuple hashing (3-tuple plus source and destination TCP or UDP ports).

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

Related Documentation
- Configuring Packet Distribution Settings on page 18
- extension-provider on page 46

license-type

Syntax license-type *license* deployment-scope [*deployments*];

Hierarchy Level [edit system extensions providers *provider-id*]

Release Information Statement introduced in Junos OS Release 10.2.
Statement introduced in Junos OS Release 11.1 for the QFX Series.

Description Configure the license type and the scope of application deployment.

Options ***license***—Type of license. Obtain correct value from the application's provider.

deployment—Scope of application deployment. You can configure a set of deployments. Obtain correct value from the application's provider.

Required Privilege Level admin—To view this statement in the configuration.
admin-control—To add this statement to the configuration.

Related Documentation
- Deploying Applications on the Router
- extensions on page 48

object-cache-size

Syntax object-cache-size *value*;

Hierarchy Level [edit chassis fpc *slot-number* pic *pic-number* adaptive-services service-package extension-provider]

Description Configure the size of the object cache. When this setting is changed, the PIC reboots.

Options ***value***—Amount of object cache, in MB. Only values in increments of 128 MB are allowed.

Range: For Multiservices 100 PIC, range is 128 MB through 512 MB. If the **wired-process-mem-size** statement at the same hierarchy level has a value of 512 MB, the maximum value for this statement is 128 MB.

Range: For Multiservices 400 PIC, range is 128 MB through 1280 MB. If the **wired-process-mem-size** statement at the same hierarchy level has a value of 512 MB, the maximum value for this statement is 512 MB.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

Related Documentation

- Configuring Memory Settings on page 19
- forwarding-db-size on page 49
- policy-db-size on page 53
- wired-process-mem-size on page 67

package (Loading on PIC)

Syntax package *package-name*;

Hierarchy Level [edit chassis fpc *slot-number* pic *pic-number* adaptive-services service-package extension-provider]

Release Information Statement introduced in Junos OS Release 9.1.

Description Identify a package to be loaded on the PIC. When a package is added or removed, the PIC reboots.

Options ***package-name***—Name of the package to be loaded on the PIC. There can be up to eight packages loaded on a PIC; however, only one data package is allowed per PIC. An error message is displayed if more than eight packages are specified.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

Related Documentation

- Configuring Packages on the PIC on page 17

package (Resource Limits)

Syntax

```
package package-name {
  resources {
    cpu {
      priority number;
      time seconds;
    }
    file {
      core-size bytes;
      open number;
      size bytes;
    }
    memory {
      data-size bytes;
      locked-in bytes;
      resident-set-size bytes;
      socket-buffers bytes;
      stack-size bytes;
    }
  }
}
```

Hierarchy Level [edit system extensions resource-limits]

Release Information Statement introduced in Junos OS Release 9.6.

Description Set resource limits for an entire package of an application.

Options *package-name*—Name of the application package.

The remaining statements are explained separately.

Required Privilege Level admin—To view this statement in the configuration.
admin-control—To add this statement to the configuration.

Related Documentation

- Restricting Resource Limits on page 26
- process on page 54
- resources on page 60
- extensions on page 48

policy-db-size

Syntax policy-db-size *size*;

Hierarchy Level [edit chassis fpc *slot-number* pic *pic-number* adaptive-services service-package extension-provider]

Description Configure the size of the policy database. When this setting is changed, the PIC reboots.

NOTE: At least one data core must be configured to configure the size of the policy database.

Options ***size***—Size of the policy database, in megabytes (MB). The size of the forwarding database and the size of the policy database together must be smaller than the size of the object cache.

Range: 0 through 1279 MB

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

Related Documentation

- Configuring Memory Settings on page 19
- forwarding-db-size on page 49
- object-cache-size on page 51
- wired-process-mem-size on page 67

process

Syntax

```
process process-ui-name {
  resources {
    cpu {
      priority number;
      time seconds;
    }
    file {
      core-size bytes;
      open number;
      size bytes;
    }
    memory {
      data-size bytes;
      locked-in bytes;
      resident-set-size bytes;
      socket-buffers bytes;
      stack-size bytes;
    }
  }
}
```

Hierarchy Level [edit system extensions resource-limits]

Release Information Statement introduced in Junos OS Release 9.6.

Description Set resource limits for a process in an application package. Limits defined for individual processes override the limits defined for an entire package.

Options *process-ui-name*—Name of the process.

The remaining statements are explained separately.

Required Privilege Level admin—To view this statement in the configuration.
admin-control—To add this statement to the configuration.

Related Documentation

- package (Resource Limits) on page 52
- resources on page 60
- extensions on page 48

process-monitor

Syntax

```
process-monitor {
  disable;
  traceoptions {
    file filename files number match regex size size (world-readable | no-world-readable);
    flag flag;
    level level;
    no-remote-trace;
  }
}
```

Hierarchy Level [edit system processes]

Release Information Statement introduced in Junos OS Release 9.0.

Description Configure tracing options for the process health monitor (pmond).

Options **disable**—Disable the health monitor process.

The remaining statements are explained separately.

Required Privilege Level admin—To view this statement in the configuration.
admin-control—To add this statement to the configuration.

Related Documentation

- traceoptions (Process Monitor) on page 63
- Verifying the Execution Status of System Processes

providers

Syntax

```
providers {
  provider-id {
    license-type license deployment-scope [ deployments ];
  }
}
```

Hierarchy Level [edit system extensions]

Release Information Statement introduced in Junos OS Release 9.0.

Description Activate the certificate of the provider of the application and enable the PIC for loading of the application.

Options

provider-id—Provider ID for the application package. The provider ID identifies the provider of the application to the system. The provider ID must be activated on the router to allow the application to be deployed on the router and run.

The remaining statements are explained separately.

Required Privilege Level

admin—To view this statement in the configuration.
admin-control—To add this statement to the configuration.

Related Documentation

- Deploying Applications on the Router

resource-cleanup

Syntax

```
resource-cleanup {
  disable;
  traceoptions {
    file filename files number match regex size size (world-readable | no-world-readable);
    flag flag;
    level level;
    no-remote-trace;
  }
}
```

Hierarchy Level [edit system processes]

Release Information Statement introduced in Junos OS Release 9.3.

Description Selectively turn on or off the debugging of trace messages for the resource cleanup process.

Options **disable**—Disable the resource cleanup process.

The remaining statements are explained separately.

Required Privilege Level trace—To view this statement in the configuration.
trace-control—To add this statement to the configuration.

Related Documentation

- traceoptions (Resource Cleanup) on page 65
- Application Health Monitoring on page 30

resource-limits

Syntax

```
resource-limits {
  package package-name {
    resources {
      cpu {
        priority number;
        time seconds;
      }
      file {
        core-size bytes;
        open number;
        size bytes;
      }
      memory {
        data-size bytes;
        locked-in bytes;
        resident-set-size bytes;
        socket-buffers bytes;
        stack-size bytes;
      }
    }
  }
  process process-ui-name {
    resources {
      cpu {
        priority number;
        time seconds;
      }
      file {
        core-size bytes;
        open number;
        size bytes;
      }
      memory {
        data-size bytes;
        locked-in bytes;
        resident-set-size bytes;
        socket-buffers bytes;
        stack-size bytes;
      }
    }
  }
}
```

Hierarchy Level [edit system extensions]

Release Information Statement introduced in Junos OS Release 9.6.

Description Set resource limits for applications using the command-line interface (CLI). You can set limits for all applications listed in the application package's manifest file or define limits for individual processes in the package. Limits defined for individual processes override the limits defined for an entire package.

The statements are explained separately.

Required Privilege Level

admin—To view this statement in the configuration.
admin-control—To add this statement to the configuration.

Related Documentation

- Restricting Resource Limits on page 26

resources

Syntax

```
resources {
  cpu {
    priority number;
    time seconds;
  }
  file {
    core-size bytes;
    open number;
    size bytes;
  }
  memory {
    data-size bytes;
    locked-in bytes;
    resident-set-size bytes;
    socket-buffers bytes;
    stack-size bytes;
  }
}
```

Hierarchy Level [edit system extensions resource-limits package *package-name*],
[edit system extensions resource-limits process *process-ui-name*]

Release Information Statement introduced in Junos OS Release 9.6.

Description Set resource limits for applications.

Options ***bytes***—Maximum size of each file, in kilobytes (KB) or megabytes (MB).

Syntax: Where **x** is some number, use **xk** to specify KB or **xm** to specify MB.

cpu—CPU resources.

- **priority *number***—Highest priority number (nice level) at which the process can run.
- **time *seconds***—Maximum amount of CPU time that can be accumulated.

file—File system resources.

- **core-size *bytes***—Maximum size of a core file that can be created.
- **open *number***—Maximum number of simultaneous open files.
- **size *bytes***—Maximum size of a file that can be created.

memory—Memory resources.

- **data-size *bytes***—Maximum size of the data segment.
- **locked-in *bytes***—Maximum number of bytes that can be locked into memory.
- **resident-set-size *bytes***—Maximum amount of private or shared memory at any given moment.

- **socket-buffers** ***bytes***—Maximum amount of physical memory that may be dedicated to the socket buffers.
- **stack-size** ***bytes***—Maximum size of the stack segment.

Required Privilege Level
admin—To view this statement in the configuration.
admin-control—To add this statement to the configuration.

Related Documentation
- Restricting Resource Limits on page 26
- resource-limits on page 58
- extensions on page 48

service-order

Syntax

```
service-order {
  forward-flow [ service-name1 service-name2 ];
  reverse-flow [ service-name1 service-name2 ];
}
```

Hierarchy Level
[edit services service-set *service-set-name*]

Release Information
Statement introduced in Junos OS Release 9.3.

Description
Define the order of services in service set to be applied to traffic coming to the PIC.

NOTE: If the extension-service statement is specified, the service-order statement is mandatory.

Options
forward-flow—Order of services in service set to be applied in forward flow.

reverse-flow—Order of services in service set to be applied in reverse flow. If you want the order to be the reverse of that specified for forward flow, this is optional. However, if, for example, you want the order to be the same regardless of direction of flow, you must include this statement. (The exception to this is for the sampling service set type. If a service set is a sampling service set and the reverse-flow service order is not configured, all sampled traffic is considered to be forward traffic.)

Required Privilege Level
system—To view this statement in the configuration.
system-control—To add this statement to the configuration.

Related Documentation
- extension-service on page 47

syslog

Syntax

```
syslog {
  facility {
    severity;
    destination destination;
  }
}
```

Hierarchy Level [edit chassis fpc *slot-number* pic *pic-number* adaptive-services service-package extension-provider]

Release Information Statement introduced in Junos OS Release 9.2.
Options **daemon** and **kernel** (for ***facility***) introduced in Junos OS Release 9.5.

Description Enable PIC system logging to record or view system log messages on a specific PIC. The system log information is passed to the kernel for logging in the **/var/log** directory.

Options ***facility***—Group of messages that are either generated by the same software process or concern a similar condition or activity. Possible values include the following: **daemon**, **external**, **kernel**, and **pfe**.

severity—Classification of effect on functioning. Possible values are the following options:

- **any**—Include all severity levels.
- **none**—Disable logging of the associated facility to a destination.
- **emergency**—System panic or other condition that causes the routing platform to stop functioning.
- **alert**—Conditions that require immediate correction, such as a corrupted system database.
- **critical**—Critical conditions, such as hard errors.
- **error**—Error conditions that generally have less serious consequences than errors in the emergency, alert, and critical levels.
- **warning**—Conditions that warrant monitoring.
- **notice**—Conditions that are not errors but might warrant special handling.
- **info**—Events or nonerror conditions of interest.

The remaining statement is explained separately.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

Related Documentation

- Configuring System Log Messages on page 21

traceoptions (Process Monitor)

Syntax

```
traceoptions {
  file filename files number match regex size size (world-readable | no-world-readable);
  flag flag;
  level level;
  no-remote-trace;
}
```

Hierarchy Level [edit system processes process-monitor]

Release Information Statement introduced in Junos OS Release 9.0.

Description Enable tracing options for the process health monitor (pmond).

Options

file *filename*—Name of the file to receive the output of the tracing operation. Enclose the name within quotation marks. To include the **file** statement, you must specify a filename.

files *number*—(Optional) Maximum number of trace files. When a trace file named **trace-file** reaches its maximum size, it is renamed **trace-file.0**, then **trace-file.1**, and so on, until the maximum number of trace files is reached. Then the oldest trace file is overwritten.

If you specify a maximum number of files, you also must specify a maximum file size with the **size** option.

flag *flag*—Specify which tracing operation to perform. To specify more than one tracing operation, include multiple **flag** statements. You can include the following flags:

- **all**—Enable all trace options flags.
- **events**—Trace process state change and cleanup events.
- **gencfg**—Trace GENCFG blobs recorded for cleanup.
- **module**—Trace module code.
- **sysvsem**—Trace SYSV semaphores recorded for cleanup.
- **sysvshm**—Trace SYSV shared memory segments recorded for cleanup.
- **tracking**—Trace tracking code.
- **ui**—Trace user interface operations.

level *level*—Specify the level of debugging output:

- **all**—Match all levels.
- **error**—Match error conditions.
- **info**—Match informational messages.
- **notice**—Match conditions that warrant special handling (but are not errors).
- **verbose**—Match verbose messages.

- **warning**—Match warning messages.

match *regex*—(Optional) Refine the output to include lines that contain the regular expression.

no-remote-trace—Disable remote tracing.

size *size*—(Optional) Maximum size of each trace file, in kilobytes (KB), megabytes (MB), or gigabytes (GB). If you specify a maximum file size, you also must specify a maximum number of trace files with the **files *number*** option.

Syntax: **xk** to specify KB, **xm** to specify MB, or **xg** to specify GB

Range: 10 KB through 1 GB

Default: 128 KB

world-readable | no-world-readable—(Optional). Grant all users permission to read log files, or restrict the permission only to the **root** user and users who have the Junos **maintenance** permission.

Required Privilege Level

trace—To view this statement in the configuration.
trace-control—To add this statement to the configuration.

Related Documentation

- Verifying the Execution Status of System Processes

traceoptions (Resource Cleanup)

Syntax

```
traceoptions {
  file filename files number match regex size size (world-readable | no-world-readable);
  flag flag;
  level level;
  no-remote-trace;
}
```

Hierarchy Level [edit system processes resource-cleanup]

Release Information Statement introduced in Junos OS Release 9.3.

Description Enable debugging tracing for resource cleanup process.

Options

file *filename*—Name of the file to receive the output of the tracing operation. Enclose the name within quotation marks. To include the **file statement**, you must specify a filename.

files *number*—(Optional) Maximum number of trace files. When a trace file named **trace-file** reaches its maximum size, it is renamed **trace-file.0**, then **trace-file.1**, and so on, until the maximum number of trace files is reached. Then the oldest trace file is overwritten.

If you specify a maximum number of files, you also must specify a maximum file size with the **size** option.

flag *flag*—Specify which tracing operation to perform. To specify more than one tracing operation, include multiple **flag** statements. You can include the following flags:

- **all**—Enable all trace options flags.
- **events**—Trace process state change and cleanup events.
- **gencfg—**—Trace GENCFG blobs recorded for cleanup.
- **module**—Trace module code.
- **sysvsem**—Trace SYSV semaphores recorded for cleanup.
- **sysvshm**—Trace SYSV shared memory segments recorded for cleanup.
- **tracking**—Trace tracking code.
- **ui**—Trace user interface operations.

level *level*—Specify the level of debugging output:

- **all**—Match all levels.
- **error**—Match error conditions.
- **info**—Match informational messages.
- **notice**—Match conditions that warrant special handling (but are not errors).
- **verbose**—Match verbose messages.

- **warning**—Match warning messages.

match *regex*—(Optional) Refine the output to include lines that contain the regular expression.

no-remote-trace—Disable remote tracing.

size *size*—(Optional) Maximum size of each trace file, in kilobytes (KB), megabytes (MB), or gigabytes (GB). If you specify a maximum file size, you also must specify a maximum number of trace files with the **files *number*** option.

Syntax: **xk** to specify KB, **xm** to specify MB, or **xg** to specify GB

Range: 10 KB through 1 GB

Default: 128 KB

world-readable | no-world-readable—(Optional). Grant all users permission to read log files, or restrict the permission only to the **root** user and users who have the Junos **maintenance** permission.

Required Privilege Level
trace—To view this statement in the configuration.
trace-control—To add this statement to the configuration.

Related Documentation
- Application Health Monitoring on page 30

wired-max-processes

Syntax
wired-max-processes *num-procs*;

Hierarchy Level
[edit chassis fpc *slot-number* pic *slot-number* adaptive-services service-package extension-provider]

Release Information
Statement introduced in Junos Release 11.4.

Description
Configure the number of processes that use wired process memory. Performance can degrade if a process uses memory beyond its Big TLB memory. If this setting is changed, the PIC will reboot.

Options
num-procs—Number of processes that use the reserved wired process memory.

Range: 1 through 8

Required Privilege Level
interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

Related Documentation
- Configuring Memory Settings on page 19
- wired-process-mem-size on page 67
- forwarding-db-size on page 49
- object-cache-size on page 51
- policy-db-size on page 53

wired-process-mem-size

Syntax wired-process-mem-size *mem-size*;

Hierarchy Level [edit chassis fpc *slot-number* pic *pic-number* adaptive-services service-package extension-provider]

Description Configure the size of the reserved wired process memory. You can also configure object cache. If this setting is changed, the PIC reboots.

Options ***megabytes***—Size of the reserved wired process memory, in MB. The only size you can set for this statement is 512 MB.

Default: 512 MB

Range: 0 through 512 MB

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

Related Documentation

- Configuring Memory Settings on page 19
- forwarding-db-size on page 49
- object-cache-size on page 51
- policy-db-size on page 53
- wired-max-processes on page 66

CHAPTER 7

Command Reference

Operational mode commands show you the current status of the router interfaces, chassis, protocols, and system information and are used to monitor and troubleshoot configurations.

The operational commands described here are relevant to the configuration of Junos SDK-built applications. For native Junos OS operational commands, see the Junos OS command references.

show extension-provider system connections

Syntax

show extension-provider system connections
<extensive>
<inet | inet6>
<interface>
<show-routing-instances>

Release Information

Command introduced in Junos OS Release 9.1.

Description

Show connection activity on the extension provider PIC. This command functions similarly to **show system connections** command.

Options

extensive—(Optional) Display exhaustive system process information.

inet | inet6—(Optional) Display IPv4 connections or IPv6 connections, respectively.

interface—(Optional) Display the name of the extension provider interface.

show-routing-instances—(Optional) Display routing instances.

Required Privilege Level

view

List of Sample Output

Output Fields

For a description of the output fields, see the output fields table for the **show system connections** command in the *Junos System Basics and Services Command Reference*.

Sample Output

show extension-provider system connections

```
user@host> show extension-provider system connections
Interface: ms-0/0/0
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  20.0.0.16.32004        20.0.0.1.64292         ESTABLISHED
tcp4       0      0  20.0.0.16.3000         20.0.0.1.58036         ESTABLISHED
tcp4       0      0  20.0.0.16.59517        20.0.0.1.3000          ESTABLISHED
tcp4       0      0  *.3000                 *.*                    LISTEN
tcp4       0      0  *.32004                *.*                    LISTEN
tcp4   66312      0  20.0.0.16.59592        20.0.0.1.32003         ESTABLISHED
tcp4       0      0  *.23                   *.*                    LISTEN
tcp4       0      0  *.33005                *.*                    LISTEN
tcp4       0      0  128.0.1.16.49900       128.0.0.1.6234         ESTABLISHED
udp4       0      0  *.842                  *.*
udp4       0      0  127.0.0.1.123          *.*
udp4       0      0  *.123                  *.*
```

```
udp46      0      0  *.514                  *.*
udp4       0      0  *.514                  *.*
Interface: ms-0/2/0
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  20.0.0.18.32004        20.0.0.1.62402         ESTABLISHED
tcp4       0      0  *.33005                *.*                    LISTEN
tcp4       0      0  128.0.3.16.59592       128.0.0.1.6234         ESTABLISHED
tcp4       0      0  *.23                   *.*                    LISTEN
tcp4       0      0  *.32004                *.*                    LISTEN
tcp4       0      0  20.0.0.18.49900        20.0.0.1.32003         ESTABLISHED
udp4       0      0  *.789                  *.*
udp4       0      0  127.0.0.1.123          *.*
udp4       0      0  *.123                  *.*
udp46      0      0  *.514                  *.*
udp4       0      0  *.514                  *.*
```

show extension-provider system connections extensive interface

```
user@host> show extension-provider system connections extensive interface ms-0/2/0
Interface: ms-0/2/0
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  20.0.0.18.32004        20.0.0.1.49792         ESTABLISHED
   sndsbcc:          0 sndsbmbcnt:          0  sndsbmbmax:     265248
sndsblowat:       2048 sndsbhiwat:      33156
   rcvsbcc:          0 rcvsbmbcnt:          0  rcvsbmbmax:     530496
rcvsblowat:          1 rcvsbhiwat:      66312
   proc id:          1  proc name:
       iss: 4025626166      sndup: 4025626167
    snduna: 4025626167     sndnxt: 4025626167      sndwnd:      66312
    sndmax: 4025626167    sndcwnd:     131070 sndssthresh: 1073725440
       irs: 3544420903      rcvup: 3544420904
    rcvnxt: 3544421176     rcvadv: 3544487488      rcvwnd:      66312
       rtt:          0       srtt:         64        rttv:         16
    rxtcur:       1200   rxtshift:          0       rtseq:          0
    rttmin:       1000  mss:       1228
     flags: REQ_SCALE RCVD_SCALE REQ_TSTMP RCVD_TSTMP SACK_PERMIT [0x20003e0]
...
```

show extension-provider system connections inet

The output for the **show extension-provider system connections inet** command is identical to that for the **show extension-provider system connections** command. For sample output, see show extension-provider system connections on page 70.

show extension-provider system connections inet6

```
user@host> show extension-provider system connections inet6
Interface: ms-0/0/0
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
udp6       0      0  *.123                  *.*
udp46      0      0  *.514                  *.*
Interface: ms-0/2/0
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
udp6       0      0  *.123                  *.*
udp46      0      0  *.514                  *.*
```

show extension-provider system connections interface

```
user@host> show extension-provider system connections interface ms-0/2/0
Interface: ms-0/2/0
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  20.0.0.18.32004        20.0.0.1.59521         ESTABLISHED
```

```
tcp4       0      0  *.33005                *.*                    LISTEN
tcp4       0      0  128.0.3.16.51534       128.0.0.1.6234         ESTABLISHED
tcp4       0      0  *.23                   *.*                    LISTEN
tcp4       0      0  *.32004                *.*                    LISTEN
tcp4       0      0  20.0.0.18.61044        20.0.0.1.32003         ESTABLISHED
udp4       0      0  *.914                  *.*
udp4       0      0  127.0.0.1.123          *.*
udp4       0      0  *.123                  *.*
udp46      0      0  *.514                  *.*
udp4       0      0  *.514                  *.*
```

show extension-provider system connections show-routing-instances

```
user@host> show extension-provider system connections show-routing-instances

Interface: ms-0/0/0
Active Internet connections (including servers) (including routing-instances)
Proto Recv-Q Send-Q Local Address Foreign Address Routing Instance  (state)
tcp4     0   0 20.0.0.16.32004  20.0.0.1.52590 __juniper_private2__ ESTABLISHED
tcp4     0   0 20.0.0.16.3000   20.0.0.1.58036 __juniper_private2__ ESTABLISHED
tcp4     0   0 20.0.0.16.59517  20.0.0.1.3000  __juniper_private2__ ESTABLISHED
tcp4     0   0 *.3000           *.*            __juniper_private2__ LISTEN
tcp4     0   0 *.32004          *.*            __juniper_private2__ LISTEN
tcp4 66312   0 20.0.0.16.59592  20.0.0.1.32003 __juniper_private2__ ESTABLISHED
tcp4     0   0 *.23             *.*            __juniper_private1__ LISTEN
tcp4     0   0 *.33005          *.*            __juniper_private2__ LISTEN
tcp4     0   0 128.0.1.16.49900 128.0.0.1.6234 __juniper_private1__ ESTABLISHED
udp4     0   0 *.842            *.*            __juniper_private1__
udp4     0   0 127.0.0.1.123    *.*            default
udp4     0   0 *.123            *.*            __juniper_private1__
udp46    0   0 *.514            *.*            default
udp4     0   0 *.514            *.*            __juniper_private1__
Interface: ms-0/2/0
Active Internet connections (including servers) (including routing-instances)
Proto Recv-Q Send-Q Local Address Foreign Address Routing Instance  (state)
tcp4     0   0 20.0.0.18.32004  20.0.0.1.54602 __juniper_private2__ ESTABLISHED
tcp4     0   0 *.33005          *.*            __juniper_private2__ LISTEN
tcp4     0   0 128.0.3.16.59592 128.0.0.1.6234 __juniper_private1__ ESTABLISHED
tcp4     0   0 *.23             *.*            __juniper_private1__ LISTEN
tcp4     0   0 *.32004          *.*            __juniper_private2__ LISTEN
tcp4     0   0 20.0.0.18.49900  20.0.0.1.32003 __juniper_private2__ ESTABLISHED
udp4     0   0 *.789            *.*            __juniper_private1__
udp4     0   0 127.0.0.1.123    *.*            default
udp4     0   0 *.123            *.*            __juniper_private1__
udp46    0   0 *.514            *.*            default
udp4     0   0 *.514            *.*            __juniper_private1__
```

show extension-provider system connections show-routing-instances interface inet6

```
user@host> show extension-provider system connections show-routing-instances interface
ms-0/0/0 inet6

Interface: ms-0/0/0
Active Internet connections (including servers) (including routing-instances)
Proto Recv-Q Send-Q  Local Address         Foreign Address       Routing Instance
      (state)
udp6       0      0  *.123                  *.*                    default
udp46      0      0  *.514                  *.*                    default
```

show extension-provider system packages

Syntax show extension-provider system packages
<detail>
<interface>

Release Information Command introduced in Junos OS Release 9.1.

Description Show packages loaded on the extension provider PIC. This command functions similarly to **show system software** command.

Options **detail**—(Optional) Display detailed output.

interface—(Optional) Display the name of the extension provider interface.

Required Privilege Level view

List of Sample Output
show extension-provider system packages on page 73
show extension-provider system packages detail on page 73
show extension-provider system packages interface on page 74

Output Fields Output field descriptions to be provided.

Sample Output

show extension-provider system packages

```
user@host> show extension-provider system packages
Interface: ms-0/0/0
jmpsdk              JUNOS MPSDK Base OS boot [9.2R1.3]
jnx-flow-data-pic   JUNOS SDK JNX-FLOW Dataplane Component [9.2I20080801_1059]
Interface: ms-0/2/0
jmpsdk              JUNOS MPSDK Base OS boot [9.2R1.3]
```

show extension-provider system packages detail

```
user@host> show extension-provider system packages detail
Interface: ms-0/0/0
Information for jmpsdk:
Comment:
JUNOS MPSDK Base OS boot [9.2R1.3]
Description:
JUNOS MPSDK Base OS
Copyright (c) 1996-2008, Juniper Networks, Inc.
All rights reserved.
Software version:       9.2R1.3
This package contains the MPSDK base operating system components.
Information for jnx-flow-data-pic:
Comment:
JUNOS SDK JNX-FLOW Dataplane Component [9.2I20080801_1059]
Description:
JUNOS SDK JNX-FLOW Data Component Package
Copyright (c) 1996-2008, Juniper Networks, Inc.
All rights reserved.
Software version:       9.2I20080801_1059
This package contains SDK JNX-FLOW dataplane component
Interface: ms-0/2/0
Information for jmpsdk:
```

```
Comment:
JUNOS MPSDK Base OS boot [9.2R1.3]
Description:
JUNOS MPSDK Base OS
Copyright (c) 1996-2008, Juniper Networks, Inc.
All rights reserved.
Software version:       9.2R1.3
This package contains the MPSDK base operating system components.
```

show extension-provider system packages interface

```
user@host> show extension-provider system packages interface ms-0/2/0
Interface: ms-0/2/0
jmpsdk              JUNOS MPSDK Base OS boot [9.2R1.3]
```

show extension-provider system processes

Syntax show extension-provider system processes
<brief | detail | extensive>
<interface>
<wide>

Release Information Command introduced in Junos OS Release 9.1.

Description Show system process table on the extension provider PIC.

Options **brief** | **detail** | **extensive**—(Optional) Display the specified level of output.

interface—(Optional) Name of the extension provider interface.

wide—(Optional) Display information even if it is wider than 80 columns.

Required Privilege Level view

List of Sample Output

Output Fields For a description of the output fields, see the output fields table for the **show system processes** command in the *Junos System Basics and Services Command Reference*.

Sample Output

show extension-provider system processes

```
user@host> show extension-provider system processes

Interface: ms-0/0/0
  PID  TT  STAT      TIME COMMAND
    0  ??  WLs    0:00.00 [swapper]
    1  ??  SLs    0:00.75 /sbin/init --
    2  ??  DL     0:05.91 [g_event]
    3  ??  DL     0:03.96 [g_up]
    4  ??  DL     0:04.24 [g_down]
    5  ??  DL     0:00.00 [kqueue taskq]
    6  ??  DL     0:00.00 [thread taskq]
    9  ??  DL     0:00.15 [pagedaemon]
   10  ??  DL     0:00.00 [ktrace]
   11  ??  RL     0:00.00 [idle: cpu31]
   12  ??  RL     0:00.00 [idle: cpu30]
   13  ??  RL     0:00.00 [idle: cpu29]
   14  ??  RL     0:00.13 [idle: cpu28]
   15  ??  RL     0:00.00 [idle: cpu27]
   16  ??  RL     0:00.00 [idle: cpu26]
   17  ??  RL     0:00.00 [idle: cpu25]
   18  ??  RL     0:00.13 [idle: cpu24]
   19  ??  RL     0:00.00 [idle: cpu23]
```

```
   20  ??  RL      0:00.00 [idle: cpu22]
   21  ??  RL      0:00.00 [idle: cpu21]
   22  ??  RL      0:00.13 [idle: cpu20]
   23  ??  RL      0:00.00 [idle: cpu19]
   24  ??  RL      0:00.00 [idle: cpu18]
   25  ??  RL      0:00.00 [idle: cpu17]
   26  ??  RL      0:00.13 [idle: cpu16]
   27  ??  RL      0:00.00 [idle: cpu15]
   28  ??  RL      0:00.00 [idle: cpu14]
   29  ??  RL      0:29.15 [idle: cpu13]
   30  ??  RL    443:15.66 [idle: cpu12]
   31  ??  RL      0:29.15 [idle: cpu11]
   32  ??  RL      0:29.14 [idle: cpu10]
. . .
```

show extension-provider system processes brief

```
user@host> show extension-provider system processes brief
Interface: ms-0/0/0
last pid: 20238;  load averages: 11.64, 11.71, 11.74  up 0+07:24:28    22:23:18
91 processes:  45 running, 34 sleeping, 12 waiting
Mem: 8924K Active, 1768K Inact, 152M Wired, 156K Cache, 77M Buf, 200M Free
Swap:
Interface: ms-0/2/0
last pid: 13025;  load averages:  4.16,  4.08,  4.02  up 0+04:43:20    22:23:18
88 processes:  37 running, 32 sleeping, 19 waiting
Mem: 6488K Active, 1212K Inact, 156M Wired, 24K Cache, 75M Buf, 460M Free
Swap:
```

show extension-provider system processes detail

```
user@host> show extension-provider system processes detail
Interface: ms-0/0/0
  PID   UID  PPID CPU PRI NI   RSS WCHAN  STARTED  TT  STAT      TIME COMMAND
   20     0     0   0 171  0     8 -       2:59PM  ??  RL     0:00.00 [idle: cp
   21     0     0   0 171  0     8 -       2:59PM  ??  RL     0:00.00 [idle: cp
   22     0     0   0 171  0     8 -       2:59PM  ??  RL     0:00.13 [idle: cp
   28     0     0   0 171  0     8 -       2:59PM  ??  RL     0:00.00 [idle: cp
   15     0     0   0 171  0     8 -       2:59PM  ??  RL     0:00.00 [idle: cp
   19     0     0   0 171  0     8 -       2:59PM  ??  RL     0:00.00 [idle: cp
   11     0     0   0 171  0     8 -       2:59PM  ??  RL     0:00.00 [idle: cp
   12     0     0   0 171  0     8 -       2:59PM  ??  RL     0:00.00 [idle: cp
   13     0     0   0 171  0     8 -       2:59PM  ??  RL     0:00.00 [idle: cp
   16     0     0   0 171  0     8 -       2:59PM  ??  RL     0:00.00 [idle: cp
   17     0     0   0 171  0     8 -       2:59PM  ??  RL     0:00.00 [idle: cp
   18     0     0   0 171  0     8 -       2:59PM  ??  RL     0:00.13 [idle: cp
   26     0     0   0 171  0     8 -       2:59PM  ??  RL     0:00.13 [idle: cp
   68     0     0   0 -16  0     8 -       2:59PM  ??  RL     0:00.00 [pot: cpu
   70     0     0   0 -16  0     8 -       2:59PM  ??  RL     0:00.00 [poller:
   14     0     0   0 171  0     8 -       2:59PM  ??  RL     0:00.13 [idle: cp
   24     0     0   0 171  0     8 -       2:59PM  ??  RL     0:00.00 [idle: cp
   25     0     0   0 171  0     8 -       2:59PM  ??  RL     0:00.00 [idle: cp
   27     0     0   0 171  0     8 -       2:59PM  ??  RL     0:00.00 [idle: cp
   69     0     0   0 -16  0     8 -       2:59PM  ??  RL     0:00.00 [poller:
   23     0     0   0 171  0     8 -       2:59PM  ??  RL     0:00.00 [idle: cp
   30     0     0   0 171  0     8 -       2:59PM  ??  RL   444:58.30 [idle: cp
  259     0     1   0 139  0   840 -       2:59PM  ??  R    3111:46.43 /opt/sdk/

  120     0     1   0 139  0   652 select  2:59PM  ??  Ss   154:48.02 syslogd -
  142     0     1   0   8  0  1132 wait    2:59PM  ??  S    122:57.26 /usr/sbin
20287     0   142   0 109  0  1100 -      10:24PM  ??  R      0:00.08 /bin/ps -
    0     0     0   0 -16  0     0 -       2:59PM  ??  WLs    0:00.00 [swapper]
```

```
    1     0     0   0   8  0  1152 wait    2:59PM  ??  SLs    0:00.75 /sbin/ini
. . .
```

show extension-provider system processes extensive

```
user@host> show extension-provider system processes extensive
Interface: ms-0/0/0
last pid: 20384;  load averages: 11.51, 11.63, 11.69  up 0+07:27:42    22:26:32
91 processes:  45 running, 34 sleeping, 12 waiting
Mem: 8924K Active, 1784K Inact, 152M Wired, 156K Cache, 77M Buf, 200M Free
Swap:
  PID USERNAME  THR PRI NICE   SIZE    RES STATE  C   TIME   WCPU COMMAND
  259 root        8 139    0   643M   840K CPU13  d  52.1H 658.40% jnx-flow-data

   22 root        1 171   52     0K     8K CPU20  14   0:00 99.22% idle: cpu20
   21 root        1 171   52     0K     8K CPU21  0    0:00 99.22% idle: cpu21
   20 root        1 171   52     0K     8K CPU22  0    0:00 99.22% idle: cpu22
   15 root        1 171   52     0K     8K CPU27  0    0:00 98.49% idle: cpu27
   19 root        1 171   52     0K     8K CPU23  0    0:00 97.71% idle: cpu23
   28 root        1 171   52     0K     8K CPU14  0    0:00 97.71% idle: cpu14
   18 root        1 171   52     0K     8K CPU24  18   0:00 96.97% idle: cpu24
   26 root        1 171   52     0K     8K CPU16  10   0:00 96.97% idle: cpu16
   70 root        1 -16    0     0K     8K CPU8   0    0:00 96.97% poller: cpu8
   68 root        1 -16    0     0K     8K CPU1   0    0:00 96.97% pot: cpu1
   25 root        1 171   52     0K     8K CPU17  0    0:00 96.97% idle: cpu17
   17 root        1 171   52     0K     8K CPU25  0    0:00 96.97% idle: cpu25
   13 root        1 171   52     0K     8K CPU29  0    0:00 96.97% idle: cpu29
   12 root        1 171   52     0K     8K CPU30  0    0:00 96.97% idle: cpu30
   11 root        1 171   52     0K     8K CPU31  0    0:00 96.97% idle: cpu31
   16 root        1 171   52     0K     8K CPU26  0    0:00 96.97% idle: cpu26
   14 root        1 171   52     0K     8K CPU28  1c   0:00 96.24% idle: cpu28
Interface: ms-0/2/0
last pid: 13175;  load averages:  4.00,  4.04,  4.00  up 0+04:46:34    22:26:32
88 processes:  37 running, 32 sleeping, 19 waiting
Mem: 6488K Active, 1228K Inact, 156M Wired, 24K Cache, 75M Buf, 460M Free
Swap:
  PID USERNAME  THR PRI NICE   SIZE    RES STATE  C   TIME   WCPU COMMAND
   12 root        1 171   52     0K     8K CPU30  0    0:00 98.49% idle: cpu30
   23 root        1 171   52     0K     8K CPU19  0    0:00 98.49% idle: cpu19
   20 root        1 171   52     0K     8K CPU22  0    0:00 98.49% idle: cpu22
   21 root        1 171   52     0K     8K CPU21  0    0:00 98.49% idle: cpu21
   75 root        1 -16    0     0K     8K CPU16  4    0:00 97.71% poller: cpu16
   76 root        1 -16    0     0K     8K CPU20  4    0:00 97.71% poller: cpu20
   13 root        1 171   52     0K     8K CPU29  0    0:00 97.71% idle: cpu29
   14 root        1 171   52     0K     8K CPU28  1c   0:00 96.24% idle: cpu28
   77 root        1 -16    0     0K     8K CPU24  4    0:00 96.24% poller: cpu24
   16 root        1 171   52     0K     8K CPU26  0    0:00 96.24% idle: cpu26
   11 root        1 171   52     0K     8K CPU31  0    0:00 96.24% idle: cpu31
   24 root        1 171   52     0K     8K CPU18  0    0:00 96.24% idle: cpu18
   25 root        1 171   52     0K     8K CPU17  0    0:00 96.24% idle: cpu17
   17 root        1 171   52     0K     8K CPU25  0    0:00 96.24% idle: cpu25
   19 root        1 171   52     0K     8K CPU23  0    0:00 96.24% idle: cpu23
   29 root        1 171   52     0K     8K CPU13  d 284:04 93.95% idle: cpu13
   30 root        1 171   52     0K     8K CPU12  c 284:05 92.48% idle: cpu12
   33 root        1 171   52     0K     8K CPU9   9 283:52 92.48% idle: cpu9
```

show extension-provider system processes interface

The output for the **show extension-provider system processes interface** command is identical to that for the **show extension-provider system processes** command except that the output for the former is for the specified interface only and the output for the latter is for all ms interfaces. For sample output, see show extension-provider system processes on page 75.

show extension-provider system processes wide

```
user@host> show extension-provider system processes wide

Interface: ms-1/0/0
  PID  TT  STAT      TIME PROVIDER COMMAND
    0  ??  WLs    0:00.00 (null)   [swapper]
    1  ??  SLs    0:00.83          /sbin/init --
    2  ??  DL     0:24.86          [g_event]
    3  ??  DL     0:24.52          [g_up]
    4  ??  DL     0:24.38          [g_down]
    5  ??  DL     0:00.00          [thread taskq]
    6  ??  DL     0:00.00          [kqueue taskq]
    9  ??  DL     0:00.53          [pagedaemon]
   10  ??  DL     0:00.00          [ktrace]
   11  ??  RL     0:00.00          [idle: cpu31]
   12  ??  RL     0:00.00          [idle: cpu30]
   13  ??  RL     0:00.00          [idle: cpu29]
   14  ??  RL     0:00.17          [idle: cpu28]
   15  ??  RL     0:00.00          [idle: cpu27]
   16  ??  RL     0:00.00          [idle: cpu26]
   17  ??  RL     0:00.00          [idle: cpu25]
   18  ??  RL     0:00.17          [idle: cpu24]
   19  ??  RL     0:00.00          [idle: cpu23]
   20  ??  RL     0:00.00          [idle: cpu22]
   21  ??  RL     0:00.00          [idle: cpu21]
   22  ??  RL     0:00.17          [idle: cpu20]
   23  ??  RL     0:00.00          [idle: cpu19]
  ...
```

show extension-provider system processes wide detail

```
user@host> show extension-provider system processes wide detail

Interface: ms-0/2/0
PID UID PPID CPU PRI NI RSS WCHAN STARTED TT STAT TIME COMMAND    PROVIDER
 12  0   0   0 171  0   8 -   5:40PM ?? RL   0:00.00 [idle: cpu30]
 20  0   0   0 171  0   8 -   5:40PM ?? RL   0:00.00 [idle: cpu22]
 23  0   0   0 171  0   8 -   5:40PM ?? RL   0:00.00 [idle: cpu19]
 21  0   0   0 171  0   8 -   5:40PM ?? RL   0:00.00 [idle: cpu21]
 25  0   0   0 171  0   8 -   5:40PM ?? RL   0:00.00 [idle: cpu17]
 11  0   0   0 171  0   8 -   5:40PM ?? RL   0:00.00 [idle: cpu31]
 13  0   0   0 171  0   8 -   5:40PM ?? RL   0:00.00 [idle: cpu29]
 14  0   0   0 171  0   8 -   5:40PM ?? RL   0:00.36 [idle: cpu28]
 16  0   0   0 171  0   8 -   5:40PM ?? RL   0:00.00 [idle: cpu26]
 17  0   0   0 171  0   8 -   5:40PM ?? RL   0:00.00 [idle: cpu25]
 19  0   0   0 171  0   8 -   5:40PM ?? RL   0:00.00 [idle: cpu23]
 24  0   0   0 171  0   8 -   5:40PM ?? RL   0:00.00 [idle: cpu18]
 75  0   0   0 -16  0   8 -   5:40PM ?? RL   0:00.00 [poller: cpu16]
 76  0   0   0 -16  0   8 -   5:40PM ?? RL   0:00.00 [poller: cpu20]
 77  0   0   0 -16  0   8 -   5:40PM ?? RL   0:00.00 [poller: cpu24]
 29  0   0   0 171  0   8 -   5:40PM ?? RL 288:22.84 [idle: cpu13]
 15  0   0   0 171  0   8 -   5:40PM ?? RL   0:00.00 [idle: cpu27]
 74  0   0   0 -16  0   8 -   5:40PM ?? RL   0:00.00 [pot: cpu1]
 33  0   0   0 171  0   8 -   5:40PM ?? RL 288:47.94 [idle: cpu9]
 30  0   0   0 171  0   8 -   5:40PM ?? RL 287:07.79 [idle: cpu12]
 34  0   0   0 171  0   8 -   5:40PM ?? RL 288:57.04 [idle: cpu8]
 37  0   0   0 171  0   8 -   5:40PM ?? RL 288:01.05 [idle: cpu5]
 38  0   0   0 171  0   8 -   5:40PM ?? RL 285:08.89 [idle: cpu4]
 42  0   0   0 171  0   8 -   5:40PM ?? RL 281:47.55 [idle: cpu0]
  0  0   0   0 -16  0   0 -   5:40PM ?? WLs  0:00.00 [swapper]    null)
  1  0   0   0   8  0 1144 wait 5:40PM ?? SLs  0:00.58 /sbin/init –
...
```

show extension-provider system uptime

Syntax show extension-provider system uptime
<interface>

Release Information Command introduced in Junos OS Release 9.1.

Description Show uptime on the extension provider PIC.

Options **interface**—(Optional) Name of the extension provider interface.

Required Privilege Level view

List of Sample Output
show extension-provider system uptime on page 79
show extension-provider system uptime interface on page 79

Output Fields For a description of the output fields, see the output fields table for the **show system uptime** command in the *Junos System Basics and Services Command Reference*.

Sample Output

show extension-provider system uptime

```
user@host> show extension-provider system uptime
Interface: ms-0/0/0
 5:08PM  up 26 mins, 0 users, load averages: 0.09, 0.06, 0.04
Interface: ms-0/2/0
 5:08PM  up 26 mins, 0 users, load averages: 0.15, 0.03, 0.01
```

show extension-provider system uptime interface

```
user@host> show extension-provider system uptime interface ms-0/2/0
Interface: ms-0/2/0
 5:08PM  up 26 mins, 0 users, load averages: 0.15, 0.03, 0.01
```

show extension-provider system virtual-memory

Syntax show extension-provider system virtual-memory
<interface>

Release Information Command introduced in Junos OS Release 9.1.

Description Show kernel dynamic memory usage on the extension provider PIC. Display the usage of Junos kernel memory listed first by size of allocation and then by type of usage.

Options **interface**—(Optional) Name of the extension provider interface.

Required Privilege Level view

List of Sample Output
show extension-provider system virtual-memory on page 80
show extension-provider system virtual-memory interface on page 81

Output Fields For a description of the output fields, see Table 4 on page 80. Output fields are listed in the approximate order in which they appear.

Table 4: show extension-provider system virtual-memory Output Fields

Field	Field Description
Size (unlabeled)	Memory block size in bytes.
Type(s) (unlabeled)	Kernel modules that are using the memory blocks. For a definition of each type, see a FreeBSD book.
interrupt	Type of interrupt: • **total**—Total number of interrupts for each type. • **rate**—Interrupt rate.
Total	Total of all interrupts.

Sample Output

show extension-provider system virtual-memory

```
user@host> show extension-provider system virtual-memory
Interface: ms-0/0/0
   916976 cpu context switches
193097320 device interrupts
    43468 software interrupts
        0 traps
   199882 system calls
       78 kernel threads created
     2416  fork() calls
        0 vfork() calls
        0 rfork() calls
        0 swap pager pageins
        0 swap pager pages paged in
        0 swap pager pageouts
        0 swap pager pages paged out
```

```
     1307 vnode pager pageins
     1307 vnode pager pages paged in
        0 vnode pager pageouts
        0 vnode pager pages paged out
        0 page daemon wakeups
        0 pages examined by the page daemon
      454 pages reactivated
    54109 copy-on-write faults
       11 copy-on-write optimized faults
    65858 zero fill pages zeroed
    65190 zero fill pages prezeroed
       10 intransit blocking page faults
   206484 total VM faults taken
        0 pages affected by kernel thread creation
   102942 pages affected by  fork()
        0 pages affected by vfork()
        0 pages affected by rfork()
   144325 pages freed
        0 pages freed by daemon
   102373 pages freed by exiting processes
     1749 pages active
      317 pages inactive
       28 pages in VM cache
    38743 pages wired down
    51895 pages free
     4096 bytes per page
        0 swap pages used
        0 peak swap pages used
   109540 total name lookups
          cache hits (86% pos + 12% neg) system 0% per-directory
          deletions 0%, falsehits 0%, toolong 0%
interrupt                total       rate
clock                177515903      59689
Total                177515903      59689
```

show extension-provider system virtual-memory interface

```
user@host> show extension-provider system virtual-memory interface ms-0/2/0
Interface: ms-0/2/0
  6971866 cpu context switches
757808764 device interrupts
   101858 software interrupts
        0 traps
  1129382 system calls
       80 kernel threads created
    15764  fork() calls
        0 vfork() calls
        0 rfork() calls
        0 swap pager pageins
        0 swap pager pages paged in
        0 swap pager pageouts
        0 swap pager pages paged out
     1212 vnode pager pageins
     1212 vnode pager pages paged in
        0 vnode pager pageouts
        0 vnode pager pages paged out
        0 page daemon wakeups
        0 pages examined by the page daemon
      420 pages reactivated
   354621 copy-on-write faults
        0 copy-on-write optimized faults
   430183 zero fill pages zeroed
   424776 zero fill pages prezeroed
```

```
     1434 intransit blocking page faults
  1332189 total VM faults taken
        0 pages affected by kernel thread creation
   648103 pages affected by  fork()
        0 pages affected by vfork()
        0 pages affected by rfork()
   892030 pages freed
        0 pages freed by daemon
   673820 pages freed by exiting processes
     1604 pages active
      309 pages inactive
        6 pages in VM cache
    39994 pages wired down
   117802 pages free
     4096 bytes per page
        0 swap pages used
        0 peak swap pages used
   702497 total name lookups
          cache hits (86% pos + 13% neg) system 0% per-directory
          deletions 0%, falsehits 0%, toolong 0%
interrupt                total       rate
clock                 987886798      47811
Total                 987886798      47811
```

show system processes health

Syntax show system processes health
<process-name *name* | pid *pid*>

Release Information Command introduced in Junos OS Release 8.5.

Description Display the resource utilization (health), of all the SDK applications currently running. You can display health information about one specific process by specifying either the process name or the process ID (PID).

Options **process-name *name***—(Optional) Display health information about the process identified by the process name.

pid *pid*—(Optional) Display health information about the process identified by the PID.

Required Privilege Level view

List of Sample Output
show system processes health on page 84
show system processes health pid on page 84
show system processes health process-name on page 85

Output Fields For a description of the output fields, see Table 5 on page 83. Output fields are listed in the approximate order in which they appear.

Table 5: show system processes health Output Fields

Field Name	Field Description
PID	Process ID, a number that identifies the process.
Provider	Provider prefix. A string that identifies the provider of the SDK application.
Parent process	Process ID of the process that spawned the process in question.
Child processes	Process ID of the process that is launched from the process in question: • Process ID for child process—A number that identifies the child process. • Name of child process—A string that identifies the child process.
CPU accumulated	Maximum amount of CPU time that can be accumulated.
Heartbeat	A regular signal sent from a router to indicate that the router is up and running: • **Interval**—Number of seconds between heartbeats. • **Allowed misses**—Number of missed heartbeats allowed before the application restarts. • **Last seen**—Time in seconds when the last heartbeat occurred. • **Total misses**—Number of heartbeats missed so far.

Table 5: show system processes health Output Fields *(continued)*

Field Name	Field Description
Resource utilization	How memory is divided: • **Area**—Segment of memory. • **Current**—Current size of memory segment. • **Max. allowed**—Maximum size allowed for memory segment. • **data size**—Current and maximum sizes of data segment. • **open files**—Number of currently open files and the maximum allowed. • **resident set size**—Current and maximum size of resident set segment. • **shared memory size**—The amount of shared memory the process is using. • **stack size**—Current and maximum sizes of stack segment.

Sample Output

show system processes health

```
user@host> show system processes health
PID: 10075 (jnx-flow-mgmt)
Provider: jnx
Parent process: 1 (init)
CPU accumulated: 0 seconds
Resource utilization:
  Area                Current  Max. allowed
  data size               7KB          32MB
  open files               20            64
  resident set size       12KB          24MB
  shared memory size       4KB
  stack size               2KB           8MB
PID: 420 (jnx-exampled)
 Provider: jnx
 Parent process: 1 (init)
 Child processes: 1
   PID    Process name
   421   jnx-exampled
CPU accumulated: 21 seconds
Heartbeat:
   Interval:       1s
   Allowed misses: 5
   Last seen:      1s ago (0 missed)
   Total misses:   2
Resource utilization:
  Area                Current  Max. allowed
  data size              24KB          16MB
  open files             23           128
  resident set size    1532KB          24MB
  shared memory size     43KB
  stack size              8KB           8MB
```

show system processes health pid

The output for the **show system processes health pid** *pid* command is identical to that for the **show system processes health** command except that health information is displayed for only the process specified by the PID. For sample output, see show system processes health on page 84.

show system processes health process-name

The output for the **show system processes health** *process-name* name command is identical to that for the **show system processes health** command except that health information is displayed for only the process named. For sample output, see show system processes health on page 84.

show system processes providers

Syntax show system processes providers

Release Information Command introduced in Junos OS Release 8.5.

Description Display information about software processes that are running on the router. The output is similar to that of the **show system processes extensive** command, but this command displays only provider processes (that is, only external processes). Also, this command's output has an extra column labeled **PROVIDER** to display the provider prefix.

Required Privilege Level view

List of Sample Output show system processes providers on page 86

Output Fields For a description of the output fields, see the output fields table for the **show system processes** command in the *Junos System Basics and Services Command Reference*. This table explains all the fields except for **PROVIDER** which is for the string that is the provider prefix for the SDK application running the process.

Sample Output

show system processes providers

```
user@host> show system processes providers
last pid:  7014;  load averages:  0.19,  0.09,  0.05  up 0+00:57:29    12:54:45
   54 processes:  1 running, 53 sleeping
   Mem: 101M Active, 105M Inact, 31M Wired, 132M Cache, 69M Buf, 369M Free
   Swap: 1536M Total, 1536M Free
     PID USERNAME PROVIDER PRI NICE   SIZE    RES STATE    TIME   WCPU COMMAND
    7001 root     jnx      96    0  3240K  2700K select   0:00  0.00% jnx-exampled
```

show system processes resource-limits process-name

Syntax show system processes resource-limits process-name *process-ui-name*

Release Information Command introduced in Junos OS Release 9.6.

Description Display the resource limits of a process in an SDK package.

Required Privilege Level view

List of Sample Output
show system processes resource-limits process-name on page 88
show system processes resource-limits process-name brief on page 88
show system processes resource-limits process-name detail on page 88

Output Fields For a description of the output fields, see Table 6 on page 87. Output fields are listed in the approximate order in which they appear.

Table 6: show system processes resource-limits process-name Output Fields

Field Name	Field Description
Provider-ID	Process ID, a number that identifies the process.
Provider-prefix	Provider prefix. A string that identifies the provider of the SDK application.
Area	Segment of memory.
Max. allowed	Current value of the limits applied in the kernel (effective limit).
Max. configurable	Maximum value the administrator can set for the limit in the configuration.
cpu/priority	Highest priority number (nice level) process can run at.
cpu/time	Maximum amount of CPU time that can be accumulated.
file/core-size	Maximum size of a core file that can be created.
file/open	Maximum number of simultaneous open files.
file/size	Maximum size of a file that can be created.
memory/data size	Maximum size of data segment.
memory/locked-in	Maximum number of bytes that can be locked into memory.
memory/resident set size	Maximum size of resident set.

Table 6: show system processes resource-limits process-name Output Fields *(continued)*

Field Name	Field Description
memory/stack size	Maximum size of stack segment.
memory/socket-buffers	Maximum amount of physical memory that may be dedicated to the socket buffers.

Sample Output

show system processes resource-limits process-name

```
user@host> show system processes resource-limits process-name jnx-example-service
Resource Limits
   Area                     Max. allowed     Max. configurable
   cpu/priority                       10                   10
   file/open                          16                   64
   memory/data-size                32MB                 32MB
   memory/locked-in                16MB                 16MB
   memory/resident-set-size         8MB                 24MB
   memory/stack-size                4MB                  8MB
```

show system processes resource-limits process-name brief

The output for the **show system processes resource-limits process-name jnx-example-service brief** command is the same as for the **show system processes resource-limits process-name jnx-example-service** command.

show system processes resource-limits process-name detail

```
user@host> show system processes resource-limits process-name jnx-example-service detail
Provider-ID:     0x8000
Provider-prefix: jnx

Resource Limits
   Area                     Max. allowed     Max. configurable
   cpu/priority                       10                   10
   cpu/time                    unlimited            unlimited
   file/size                   unlimited            unlimited
   file/open                          16                   64
   file/core-size              unlimited            unlimited
   memory/data-size                32MB                 32MB
   memory/locked-in                16MB                 16MB
   memory/resident-set-size         8MB                 24MB
   memory/stack-size                4MB                  8MB
   memory/socket-buffers       unlimited            unlimited
```

show system resource-cleanup processes

Syntax

show system resource-cleanup processes
<detail>
<pid *number*>
<process-name *name*>

Release Information

Command introduced in Junos OS Release 9.3.
Command introduced in Junos OS Release 11.1 for the QFX Series.

Description

Display the list of processes that have been registered for resource cleanup services.

Options

detail—(Optional) Display the list of processes that have been registered for resource cleanup services, along with the resources that have been requested for cleanup.

pid *number*—(Optional) Display a process that has been registered for resource cleanup services by specifying the Process Identifier number.

process-name *name*—(Optional) Display a process that has been registered for resource cleanup services by name of the process.

Required Privilege Level

view

List of Sample Output

show system resource-cleanup processes on page 89
show system resource-cleanup processes detail on page 89

Output Fields

For a description of the output fields, see Table 7 on page 89. Output fields are listed in the approximate order in which they appear.

Table 7: show system resource-cleanup processes Output Fields

Field Name	Field Description
PID	Process ID, a number that identifies a process.
Process name	String that identifies the process.
Resources to clean	Resources that have been registered to be cleaned up.

Sample Output

show system resource-cleanup processes

```
user@host> show system resource-cleanup processes
PID     Process name          Resources to clean
420     jnx-exampled          GENCFG, SYSV shared memory
```

show system resource-cleanup processes detail

```
user@host> show system resource-cleanup processes detail
PID     Process name          Resources to clean
420     jnx-exampled          GENCFG blob major ID 0x8000, minor ID 0x0000
```

```
SYSV shared memory ID 65536, key 1108955839
SYSV shared memory ID 65537, key 1108955837
```

CHAPTER 8

Indexes

Index

Symbols

A

B

C

D

E

F

H

L

M

Index of Statements and Commands