



Junos[®] OS

Routing Protocols Overview

Release
12.1



Published: 2012-03-13

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

This product includes the Envoy SNMP Engine, developed by Epilogue Technology, an Integrated Systems Company. Copyright © 1986-1997, Epilogue Technology Corporation. All rights reserved. This program and its documentation were developed at private expense, and no part of them is in the public domain.

This product includes memory allocation software developed by Mark Moraes, copyright © 1988, 1989, 1993, University of Toronto.

This product includes FreeBSD software developed by the University of California, Berkeley, and its contributors. All of the documentation and software included in the 4.4BSD and 4.4BSD-Lite Releases is copyrighted by the Regents of the University of California. Copyright © 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994. The Regents of the University of California. All rights reserved.

GateD software copyright © 1995, the Regents of the University. All rights reserved. Gate Daemon was originated and developed through release 3.0 by Cornell University and its collaborators. Gated is based on Kirton's EGP, UC Berkeley's routing daemon (routed), and DCN's HELLO routing protocol. Development of Gated has been supported in part by the National Science Foundation. Portions of the GateD software copyright © 1988, Regents of the University of California. All rights reserved. Portions of the GateD software copyright © 1991, D. L. S. Associates.

This product includes software developed by Maker Communications, Inc., copyright © 1996, 1997, Maker Communications, Inc.

Juniper Networks, Junos, Steel-Belted Radius, NetScreen, and ScreenOS are registered trademarks of Juniper Networks, Inc. in the United States and other countries. The Juniper Networks Logo, the Junos logo, and JunosE are trademarks of Juniper Networks, Inc. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners.

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

Products made or sold by Juniper Networks or components thereof might be covered by one or more of the following patents that are owned by or licensed to Juniper Networks: U.S. Patent Nos. 5,473,599, 5,905,725, 5,909,440, 6,192,051, 6,333,650, 6,359,479, 6,406,312, 6,429,706, 6,459,579, 6,493,347, 6,538,518, 6,538,899, 6,552,918, 6,567,902, 6,578,186, and 6,590,785.

Junos® OS Routing Protocols Overview

12.1

Copyright © 2012, Juniper Networks, Inc.
All rights reserved.

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

YEAR 2000 NOTICE

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

END USER LICENSE AGREEMENT

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

Table of Contents

	About the Documentation	ix
	Documentation and Release Notes	ix
	Supported Platforms	ix
	Using the Examples in This Manual	ix
	Merging a Full Example	x
	Merging a Snippet	x
	Documentation Conventions	xi
	Documentation Feedback	xiii
	Requesting Technical Support	xiii
	Self-Help Online Tools and Resources	xiii
	Opening a Case with JTAC	xiv
Part 1	Overview	
Chapter 1	Introduction to Routing Protocols	3
	Routing Databases Overview	3
	Routing Protocol Databases	3
	Junos Routing Tables	4
	Forwarding Tables	4
	How the Routing and Forwarding Tables Are Synchronized	5
	Route Preferences Overview	6
	Alternate and Tiebreaker Preferences	6
	Multiple Active Routes	6
	Equal-Cost Paths and Load Sharing	7
	IPv6 Overview	8
	IPv6 Packet Headers	8
	Header Structure	9
	Extension Headers	9
	IPv6 Addressing	9
	Address Representation	10
	Address Types	10
	Address Scope	10
	Address Structure	11
Part 2	Administration	
Chapter 2	Routing Protocols Reference	15
	Understanding BGP Path Selection	15
	Understanding Route Preference Values	18
	IPv6 Standards	19

Chapter 3	Complete Routing and Routing Protocol Configuration Statements	21
	[edit logical-systems] Hierarchy Level	21
	[edit protocols] Hierarchy Level	22
	[edit routing-instances] Hierarchy Level	39
	[edit routing-options] Hierarchy Level	44
Part 3	Troubleshooting	
Chapter 4	Routing Protocol Process Memory FAQ	53
	Routing Protocol Process Memory FAQ Overview	53
	Routing Protocol Process Memory FAQs	54
	Routing Protocol Process Memory Utilization FAQs	54
	Interpreting Routing Protocol Process-Related Command Outputs FAQs	56
	Routing Protocol Process Memory Swapping FAQs	59
	Troubleshooting the Routing Protocol Process FAQs	60
Part 4	Index	
	Index	63

List of Figures

Part 1	Overview	
Chapter 1	Introduction to Routing Protocols	3
	Figure 1: Synchronizing Routing Exchange Between the Routing and Forwarding Tables	5

List of Tables

	About the Documentation	ix
	Table 1: Notice Icons	xi
	Table 2: Text and Syntax Conventions	xi
Part 2	Administration	
Chapter 2	Routing Protocols Reference	15
	Table 3: Default Route Preference Values	18
Part 3	Troubleshooting	
Chapter 4	Routing Protocol Process Memory FAQ	53
	Table 4: show system processes extensive Output Fields	57
	Table 5: show task memory Output Fields	58

About the Documentation

- Documentation and Release Notes on page ix
- Supported Platforms on page ix
- Using the Examples in This Manual on page ix
- Documentation Conventions on page xi
- Documentation Feedback on page xiii
- Requesting Technical Support on page xiii

Documentation and Release Notes

To obtain the most current version of all Juniper Networks[®] technical documentation, see the product documentation page on the Juniper Networks website at <http://www.juniper.net/techpubs/>.

If the information in the latest release notes differs from the information in the documentation, follow the product Release Notes.

Juniper Networks Books publishes books by Juniper Networks engineers and subject matter experts. These books go beyond the technical documentation to explore the nuances of network architecture, deployment, and administration. The current list can be viewed at <http://www.juniper.net/books>.

Supported Platforms

For the features described in this document, the following platforms are supported:

- T Series
- MX Series
- M Series

Using the Examples in This Manual

If you want to use the examples in this manual, you can use the **load merge** or the **load merge relative** command. These commands cause the software to merge the incoming configuration into the current candidate configuration. The example does not become active until you commit the candidate configuration.

If the example configuration contains the top level of the hierarchy (or multiple hierarchies), the example is a *full example*. In this case, use the **load merge** command.

If the example configuration does not start at the top level of the hierarchy, the example is a *snippet*. In this case, use the **load merge relative** command. These procedures are described in the following sections.

Merging a Full Example

To merge a full example, follow these steps:

1. From the HTML or PDF version of the manual, copy a configuration example into a text file, save the file with a name, and copy the file to a directory on your routing platform.

For example, copy the following configuration to a file and name the file **ex-script.conf**. Copy the **ex-script.conf** file to the **/var/tmp** directory on your routing platform.

```
system {
  scripts {
    commit {
      file ex-script.xml;
    }
  }
}
interfaces {
  fxp0 {
    disable;
    unit 0 {
      family inet {
        address 10.0.0.1/24;
      }
    }
  }
}
```

2. Merge the contents of the file into your routing platform configuration by issuing the **load merge** configuration mode command:

```
[edit]
user@host# load merge /var/tmp/ex-script.conf
load complete
```

Merging a Snippet

To merge a snippet, follow these steps:

1. From the HTML or PDF version of the manual, copy a configuration snippet into a text file, save the file with a name, and copy the file to a directory on your routing platform.

For example, copy the following snippet to a file and name the file **ex-script-snippet.conf**. Copy the **ex-script-snippet.conf** file to the **/var/tmp** directory on your routing platform.

```
commit {
  file ex-script-snippet.xml; }
```

2. Move to the hierarchy level that is relevant for this snippet by issuing the following configuration mode command:

```
[edit]
user@host# edit system scripts
[edit system scripts]
```

3. Merge the contents of the file into your routing platform configuration by issuing the **load merge relative** configuration mode command:

```
[edit system scripts]
user@host# load merge relative /var/tmp/ex-script-snippet.conf
load complete
```

For more information about the **load** command, see the [Junos OS CLI User Guide](#).

Documentation Conventions

Table 1 on page xi defines notice icons used in this guide.

Table 1: Notice Icons





Icon	Meaning	Description
	Informational note	Indicates important features or instructions.
	Caution	Indicates a situation that might result in loss of data or hardware damage.
	Warning	Alerts you to the risk of personal injury or death.
	Laser warning	Alerts you to the risk of personal injury from a laser.

Table 2 on page xi defines the text and syntax conventions used in this guide.

Table 2: 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

Table 2: Text and Syntax Conventions (*continued*)

Convention	Description	Examples
<i>Italic text like this</i>	<ul style="list-style-type: none"> Introduces important new terms. Identifies book names. Identifies RFC and Internet draft titles. 	<ul style="list-style-type: none"> A policy <i>term</i> is a named structure that defines match conditions and actions. <i>Junos OS System Basics Configuration Guide</i> RFC 1997, <i>BGP Communities Attribute</i>
<i>Italic text like this</i>	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 <i>domain-name</i>
Text like this	Represents names of configuration statements, commands, files, and directories; interface names; configuration hierarchy levels; or labels on routing platform components.	<ul style="list-style-type: none"> 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 <i>metric</i> >;
(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 (<i>string1</i> <i>string2</i> <i>string3</i>)
# (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 [<i>community-ids</i>]
Indentation and braces ({ })	Identify a level in the configuration hierarchy.	[edit] routing-options { static { route default { nexthop <i>address</i> ; 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.	<ul style="list-style-type: none"> 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> .

PART 1

Overview

- [Introduction to Routing Protocols on page 3](#)

CHAPTER 1

Introduction to Routing Protocols

- [Routing Databases Overview on page 3](#)
- [Route Preferences Overview on page 6](#)
- [Equal-Cost Paths and Load Sharing on page 7](#)
- [IPv6 Overview on page 8](#)

Routing Databases Overview

The Junos OS maintains two databases for routing information:

- **Routing table**—Contains all the routing information learned by all routing protocols.
- **Forwarding table**—Contains the routes actually used to forward packets through the router.

In addition, the interior gateway protocols (IGPs), IS-IS, and OSPF maintain link-state databases.

This section includes the following topics:

- [Routing Protocol Databases on page 3](#)
- [Junos Routing Tables on page 4](#)
- [Forwarding Tables on page 4](#)
- [How the Routing and Forwarding Tables Are Synchronized on page 5](#)

Routing Protocol Databases

Each IGP routing protocol maintains a database of the routing information it has learned from other routers running the same protocol and uses this information as defined and required by the protocol. IS-IS and OSPF use the routing information they received to maintain link-state databases, which they use to determine which adjacent neighbors are operational and to construct network topology maps.

IS-IS and OSPF use the Dijkstra algorithm, and RIP and RIPv6 use the Bellman-Ford algorithm to determine the best route or routes (if there are multiple equal-cost routes) to reach each destination and install these routes into the Junos OS routing table.

When you configure a protocol on an interface, you must also configure a protocol family on that interface.

Junos Routing Tables

The Junos OS routing table is used by the routing protocol process to maintain its database of routing information. In this table, the routing protocol process stores statically configured routes, directly connected interfaces (also called *direct routes* or *interface routes*), and all routing information learned from all routing protocols. The routing protocol process uses this collected routing information to select the *active route* to each destination, which is the route that actually is used to forward packets to that destination.

By default, the Junos OS maintains three routing tables: one for unicast routes, another for multicast routes, and a third for MPLS. You can configure additional routing tables to support situations where you need to separate a particular group of routes or where you need greater flexibility in manipulating routing information. In general, most operations can be performed without resorting to the complexity of additional routing tables. However, creating additional routing tables has several specific uses, including importing interface routes into more than one routing table, applying different routing policies when exporting the same route to different peers, and providing greater flexibility with incongruent multicast topologies.

Each routing table is identified by a name, which consists of the protocol family followed by a period and a small, nonnegative integer. The protocol family can be **inet** (Internet), **iso** (ISO), or **mpls** (MPLS). The following names are reserved for the default routing tables maintained by the Junos OS:

- **inet.0**—Default IP version 4 (IPv4) unicast routing table
- **inet6.0**—Default IP version 6 (IPv6) unicast routing table
- **instance-name.inet.0**—Unicast routing table for a particular routing instance
- **inet.1**—Multicast forwarding cache
- **inet.2**—Unicast routes used for multicast reverse path forwarding (RPF) lookup
- **inet.3**—MPLS routing table for path information
- **mpls.0**—MPLS routing table for label-switched path (LSP) next hops



NOTE: For clarity, this manual contains general discussions of routing tables as if there were only one table. However, when it is necessary to distinguish among the routing tables, their names are explicitly used.

Forwarding Tables

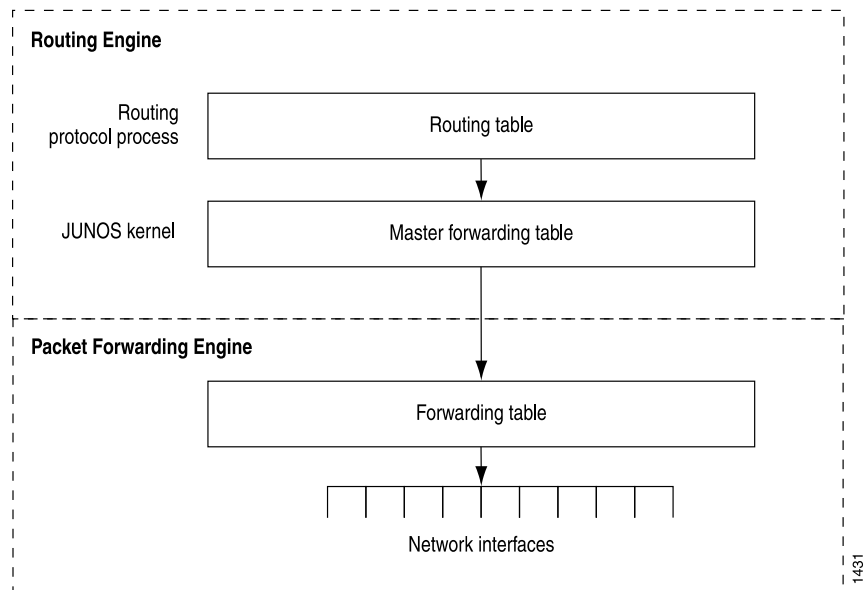
The Junos OS installs all active routes from the routing table into the forwarding table. The active routes are used to forward packets to their destinations.

The Junos OS kernel maintains a master copy of the forwarding table. It copies the forwarding table to the Packet Forwarding Engine, which is the part of the router responsible for forwarding packets.

How the Routing and Forwarding Tables Are Synchronized

The Junos OS routing protocol process is responsible for synchronizing the routing information between the routing and forwarding tables. To do this, the routing protocol process calculates the active routes from all the routes in the routing table and installs them into the forwarding table. The routing protocol process then copies the forwarding table to the router's Packet Forwarding Engine, the part of the router that forwards packets. [Figure 1 on page 5](#) illustrates how the routing tables are synchronized.

Figure 1: Synchronizing Routing Exchange Between the Routing and Forwarding Tables



Route Preferences Overview

For unicast routes, the Junos OS routing protocol process uses the information in its routing table, along with the properties set in the configuration file, to choose an *active route* for each destination. While the Junos OS might know of many routes to a destination, the active route is the preferred route to that destination and is the one that is installed in the forwarding table and used when actually routing packets.

The routing protocol process generally determines the active route by selecting the route with the lowest preference value. The preference value is an arbitrary value in the range from 0 through 4,294,967,295 ($2^{32} - 1$) that the software uses to rank routes received from different protocols, interfaces, or remote systems.

The preference value is used to select routes to destinations in external autonomous systems (ASs) or routing domains; it has no effect on the selection of routes within an AS (that is, within an interior gateway protocol [IGP]). Routes within an AS are selected by the IGP and are based on that protocol's metric or cost value.

This section includes the following topics:

- [Alternate and Tiebreaker Preferences on page 6](#)
- [Multiple Active Routes on page 6](#)

Alternate and Tiebreaker Preferences

The Junos OS provides support for alternate and tiebreaker preferences, and some of the routing protocols, including BGP and label switching, use these additional preferences. With these protocols, you can specify a primary route preference (by including the **preference** statement in the configuration), and a secondary preference that is used as a tiebreaker (by including the **preference2** statement). You can also mark route preferences with additional route tiebreaker information by specifying a color and a tiebreaker color (by including the **color** and **color2** statements in the configuration).

The software uses a 4-byte value to represent the route preference value. When using the preference value to select an active route, the software first compares the primary route preference values, choosing the route with the lowest value. If there is a tie and a secondary preference has been configured, the software compares the secondary preference values, choosing the route with the lowest value. The secondary preference values must be included in a set for the preference values to be considered.

Multiple Active Routes

The IGPs compute equal-cost multipath next hops, and IBGP picks up these next hops. When there are multiple, equal-cost next hops associated with a route, the routing protocol process installs only one of the next hops in the forwarding path with each route, randomly selecting which next hop to install. For example, if there are 3 equal-cost paths to an exit routing device and 900 routes leaving through that routing device, each path ends up with about 300 routes pointing at it. This mechanism provides load distribution among the paths while maintaining packet ordering per destination.

BGP multipath does not apply to paths that share the same MED-plus-IGP cost yet differ in IGP cost. Multipath path selection is based on the IGP cost metric, even if two paths have the same MED-plus-IGP cost.

Equal-Cost Paths and Load Sharing

For equal-cost paths, load sharing is based on the BGP next hop. For example, if four prefixes all point to a next hop and there is more than one equal-cost path to that next hop, the routing protocol process uses a hash algorithm to choose the path among the four prefixes. Also, for each prefix, the routing protocol process installs a single forwarding entry pointing along one of the paths. The routing software does not rehash the path taken as prefixes pointing to the next hop come and go, but it does rehash if the number of paths to the next hop changes. Because a prefix is tied to a particular path, packet reordering should not happen. The degree of load sharing improves as the number of prefixes increases.

IPv6 Overview

IP version 6 (IPv6) is the latest version of IP. IP enables numerous nodes on different networks to interoperate seamlessly. IP version 4 (IPv4) is currently used in intranets and private networks, as well as the Internet. IPv6 is the successor to IPv4, and is based for the most part on IPv4.

IPv4 has been widely deployed and used to network the Internet today. With the rapid growth of the Internet, enhancements to IPv4 are needed to support the influx of new subscribers, Internet-enabled devices, and applications. IPv6 is designed to enable the global expansion of the Internet.

IPv6 builds upon the functionality of IPv4, providing improvements to addressing, configuration and maintenance, and security.

IPv6 offers the following benefits:

- Expanded addressing capabilities—IPv6 provides a larger address space. IPv6 addresses consist of 128 bits, while IPv4 addresses consist of 32 bits. 128-bit addressing increases the address space by approximately 1029 unique addresses, enough to last for the foreseeable future.
- Header format simplification—IPv6 packet header format is designed to be efficient. IPv6 standardizes the size of the packet header to 40 bytes, divided into 8 fields.
- Improved support for extensions and options—Extension headers carry Internet-layer information and have a standard size and structure.
- Flow labeling capability—Flow labels provide consistent handling of packets belonging to the same flow.
- Improved privacy and security—IPv6 supports extensions for authentication and data integrity, which enhance privacy and security.

This section discusses the following topics:

- [IPv6 Packet Headers on page 8](#)
- [IPv6 Addressing on page 9](#)

IPv6 Packet Headers

IPv6 headers are different from IPv4 headers.

This section discusses the following topics that provide background information about IPv6 headers:

- [Header Structure on page 9](#)
- [Extension Headers on page 9](#)

Header Structure

IPv6 packet headers contain many of the fields found in IPv4 packet headers; some of these fields have been modified from IPv4. The 40-byte IPv6 header consists of the following 8 fields:

- Traffic class—Class-of-service (CoS) priority of the packet. Previously the type-of-service (ToS) field in IPv4. However, the semantics of this field (for example, DiffServ code points) are identical to IPv4.
- Destination address—Final destination node address for the packet.
- Flow label—Packet flows requiring a specific class of service. The flow label identifies all packets belonging to a specific flow, and routers can identify these packets and handle them in a similar fashion.
- Hop limit—Maximum number of hops allowed. Previously the time-to-live (TTL) field in IPv4.
- Next header—Next extension header to examine. Previously the protocol field in IPv4.
- Payload length—Length of the IPv6 payload. Previously the total length field in IPv4.
- Source address—Address of the source node sending the packet.
- Version—Version of IP.

Extension Headers

In IPv6, *extension headers* are used to encode optional Internet-layer information.

Extension headers are placed between the IPv6 header and the upper layer header in a packet.

Extension headers are chained together using the next header field in the IPv6 header. The next header field indicates to the router which extension header to expect next. If there are no more extension headers, the next header field indicates the upper layer header (TCP header, User Datagram Protocol [UDP] header, ICMPv6 header, an encapsulated IP packet, or other items).

IPv6 Addressing

IPv6 uses a 128-bit addressing model. This creates a much larger address space than IPv4 addresses, which are made up of 32 bits. IPv6 addresses also contain a scope field that categorizes what types of applications are suitable for the address. IPv6 does not support broadcast addresses, but instead uses multicast addresses to serve this role. In addition, IPv6 also defines a new type of address called *anycast*.



NOTE: You cannot configure a subnet zero IPv6 address because RFC 2461 reserves the subnet-zero address for anycast addresses, and Junos OS complies with the RFC.

This section discusses the following topics that provide background information about IPv6 addressing:

- [Address Representation on page 10](#)
- [Address Types on page 10](#)
- [Address Scope on page 10](#)
- [Address Structure on page 11](#)

Address Representation

IPv6 addresses consist of 8 groups of 16-bit hexadecimal values separated by colons (:). The IPv6 address format is as follows:

`aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa`

aaaa is a 16-bit hexadecimal value, and **a** is a 4-bit hexadecimal value. Following is an example of an actual IPv6 address:

`3FFE:0000:0000:0001:0200:F8FF:FE75:50DF`

You can omit the leading zeros, as shown:

`3FFE:0:0:1:200:F8FF:FE75:50DF`

You can compress 16-bit groups of zeros to the notation `::` (two colons), as shown here, but only once per address:

`3FFE::1:200:F8FF:FE75:50DF`

Address Types

There are three types of IPv6 addresses:

- Unicast—For a single interface.
- Multicast—For a set of interfaces on the same physical medium. A packet is sent to all of the interfaces associated with the address.
- Anycast—For a set of interfaces on different physical mediums. A packet is sent to only one of the interfaces associated with this address, not to all the interfaces.

Address Scope

IPv6 addresses have *scope*, which identifies the application suitable for the address. Unicast and multicast addresses support scoping.

Unicast addresses support two types of scope: *global* scope and *local* scope. There are two types of local scope: *link-local* addresses and *site-local* addresses. Link-local unicast addresses are used within a single network link. The first ten bits of the prefix identify the address as a link-local address. Link-local addresses cannot be used outside a network link. Site-local unicast addresses are used within a site or intranet. A site consists of multiple network links, and site-local addresses identify nodes inside the intranet. Site-local addresses cannot be used outside the site.

Multicast addresses support 16 different types of scope, including node, link, site, organization, and global scope. A 4-bit field in the prefix identifies the scope.

Address Structure

Unicast addresses identify a single interface. The address consists of n bits for the prefix, and $128 - n$ bits for the interface ID.

Multicast addresses identify a set of interfaces. The address is made up of the first 8 bits of all ones, a 4-bit flags field, a 4-bit scope field, and a 112-bit group ID:

`11111111 | flags | scope | group ID`

The first octet of ones identifies the address as a multicast address. The flags field identifies whether the multicast address is a well-known address or a transient multicast address. The scope field identifies the scope of the multicast address. The 112-bit group ID identifies the multicast group.

Similar to multicast addresses, anycast addresses identify a set of interfaces. However, packets are sent to only one of the interfaces, not to all interfaces. Anycast addresses are allocated from the normal unicast address space and cannot be distinguished from a unicast address in format. Therefore, each member of an anycast group must be configured to recognize certain addresses as anycast addresses.

PART 2

Administration

- [Routing Protocols Reference on page 15](#)
- [Complete Routing and Routing Protocol Configuration Statements on page 21](#)

CHAPTER 2

Routing Protocols Reference

- [Understanding BGP Path Selection on page 15](#)
- [Understanding Route Preference Values on page 18](#)
- [IPv6 Standards on page 19](#)

Understanding BGP Path Selection

For each prefix in the routing table, the routing protocol process selects a single best path. After the best path is selected, the route is installed in the routing table. The best path becomes the active route if the same prefix is not learned by a protocol with a lower (more preferred) global preference value, also known as the administrative distance. The algorithm for determining the active route is as follows:

1. Verify that the next hop can be resolved.
2. Choose the path with the lowest preference value (routing protocol process preference).

Routes that are not eligible to be used for forwarding (for example, because they were rejected by routing policy or because a next hop is inaccessible) have a preference of -1 and are never chosen.

3. Prefer the path with higher local preference.

For non-BGP paths, choose the path with the lowest **preference2** value.

4. If the accumulated interior gateway protocol (AIGP) attribute is enabled, prefer the path with the lower AIGP attribute.
5. Prefer the path with the shortest autonomous system (AS) path value (skipped if the **as-path-ignore** statement is configured).

A confederation segment (sequence or set) has a path length of 0. An AS set has a path length of 1.

6. Prefer the route with the lower origin code.

Routes learned from an IGP have a lower origin code than those learned from an exterior gateway protocol (EGP), and both have lower origin codes than incomplete routes (routes whose origin is unknown).

7. Prefer the path with the lowest multiple exit discriminator (MED) metric.

Depending on whether nondeterministic routing table path selection behavior is configured, there are two possible cases:

- If nondeterministic routing table path selection behavior is not configured (that is, if the **path-selection cisco-nondeterministic** statement is not included in the BGP configuration), for paths with the same neighboring AS numbers at the front of the AS path, prefer the path with the lowest MED metric. To always compare MEDs whether or not the peer ASs of the compared routes are the same, include the **path-selection always-compare-med** statement.
- If nondeterministic routing table path selection behavior is configured (that is, the **path-selection cisco-nondeterministic** statement is included in the BGP configuration), prefer the path with the lowest MED metric.

Confederations are not considered when determining neighboring ASs. A missing MED metric is treated as if a MED were present but zero.



NOTE: MED comparison works for single path selection within an AS (when the route does not include an AS path), though this usage is uncommon.

8. Prefer strictly internal paths, which include IGP routes and locally generated routes (static, direct, local, and so forth).
9. Prefer strictly external BGP (EBGP) paths over external paths learned through internal BGP (IBGP) sessions.
10. Prefer the path whose next hop is resolved through the IGP route with the lowest metric.



NOTE: A path is considered a BGP equal-cost path (and will be used for forwarding) if a tie-break is performed after the previous step. All paths with the same neighboring AS, learned by a multipath-enabled BGP neighbor, are considered.

BGP multipath does not apply to paths that share the same MED-plus-IGP cost yet differ in IGP cost. Multipath path selection is based on the IGP cost metric, even if two paths have the same MED-plus-IGP cost.

11. If both paths are external, prefer the currently active path to minimize route-flapping. This rule is not used if:
 - **path-selection external-router-id** is configured.
 - Both peers have the same router ID.
 - Either peer is a confederation peer.
 - Neither path is the current active path.

12. Prefer the path from the peer with the lowest router ID. For any path with an originator ID attribute, substitute the originator ID for the router ID during router ID comparison.
13. Prefer the path with the shortest cluster list length. The length is 0 for no list.
14. Prefer the path from the peer with the lowest peer IP address.

By default, only the multiple exit discriminators (MEDs) of routes that have the same peer autonomous systems (ASs) are compared. You can configure routing table path selection options to obtain different behaviors.

The third step of the algorithm, by default, evaluates the length of the AS path and determines the active path. You can configure an option that enables Junos OS to skip this third step of the algorithm by including the **as-path-ignore** option.



NOTE: The **as-path-ignore** option is not supported for routing instances.

To configure routing table path selection behavior, include the **path-selection** statement:

```
path-selection {
  (always-compare-med | cisco-non-deterministic | external-router-id);
  as-path-ignore;
  med-plus-igp {
    igp-multiplier number;
    med-multiplier number;
  }
}
```

For a list of hierarchy levels at which you can include this statement, see the statement summary section for this statement.

Routing table path selection can be configured in one of the following ways:

- Using the same nondeterministic behavior as does the Cisco IOS software (**cisco-non-deterministic**). This behavior has two effects:
 - The active path is always first. All nonactive but eligible paths follow the active path and are maintained in the order in which they were received, with the most recent path first. Ineligible paths remain at the end of the list.
 - When a new path is added to the routing table, path comparisons are made without removing from consideration those paths that should never be selected because those paths lose the MED tie-breaking rule.



NOTE: The result of these two effects is that the system only sometimes compares the MED values between paths that it should otherwise compare. Because of this, we recommend that you not configure nondeterministic behavior.

- Always comparing MEDs whether or not the peer ASs of the compared routes are the same (**always-compare-med**).

- Comparing the router ID between external BGP paths to determine the active path (**external-router-id**). By default, router ID comparison is not performed if one of the external paths is active. You can force the router ID comparison by restarting the routing process with the **restart routing** operational-mode command.
- Adding the IGP cost to the next-hop destination to the MED value before comparing MED values for path selection.

BGP multipath does not apply to paths that share the same MED-plus-IGP cost, yet differ in IGP cost. Multipath path selection is based on the IGP cost metric, even if two paths have the same MED-plus-IGP cost.

Related Documentation

- Example: Ignoring the AS Path Attribute When Selecting the Best Path
- Example: Always Comparing MEDs

Understanding Route Preference Values

The Junos OS routing protocol process assigns a default preference value (also known as an administrative distance) to each route that the routing table receives. The default value depends on the source of the route. The preference value is a value from 0 through 4,294,967,295 ($2^{32} - 1$), with a lower value indicating a more preferred route.

[Table 3 on page 18](#) lists the default preference values.

Table 3: Default Route Preference Values

How Route Is Learned	Default Preference	Statement to Modify Default Preference
Directly connected network	0	—
System routes	4	—
Static and Static LSPs	5	static
RSVP-signaled LSPs	7	RSVP preference as described in the <i>Junos OS MPLS Applications Configuration Guide</i>
LDP-signaled LSPs	9	LDP preference , as described in the <i>Junos OS MPLS Applications Configuration Guide</i>
OSPF internal route	10	OSPF preference
IS-IS Level 1 internal route	15	IS-IS preference
IS-IS Level 2 internal route	18	IS-IS preference
Redirects	30	—
Kernel	40	—

Table 3: Default Route Preference Values (*continued*)

How Route Is Learned	Default Preference	Statement to Modify Default Preference
SNMP	50	–
Router discovery	55	–
RIP	100	RIP preference
RIPng	100	RIPng preference
PIM	105	<i>Junos OS Multicast Protocols Configuration Guide</i>
DVMRP	110	<i>Junos OS Multicast Protocols Configuration Guide</i>
Aggregate	130	aggregate
OSPF AS external routes	150	OSPF external-preference
IS-IS Level 1 external route	160	IS-IS external-preference
IS-IS Level 2 external route	165	IS-IS external-preference
BGP	170	BGP preference, export, import
MSDP	175	<i>Junos OS Multicast Protocols Configuration Guide</i>

In general, the narrower the scope of the statement, the higher precedence its preference value is given, but the smaller the set of routes it affects. To modify the default preference value for routes learned by routing protocols, you generally apply routing policy when configuring the individual routing protocols. You also can modify some preferences with other configuration statements, which are indicated in the table.

Related Documentation

- [Junos OS Policy Framework Configuration Guide](#)

IPv6 Standards

IPv6 is defined in the following documents:

- RFC 1981, *Path MTU Discovery for IP version 6*
- RFC 2373, *IP Version 6 Addressing Architecture*
- RFC 2460, *Internet Protocol, Version 6 (IPv6)*
- RFC 2461, *Neighbor Discovery for IP Version 6*
- RFC 2462, *IPv6 Stateless Address Auto configuration*
- RFC 2463, *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6*

- RFC 2464, *Transmission of IPv6 Packets over Ethernet Networks*
- RFC 2472, *IP Version 6 over PPP*
- RFC 2474, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*
- RFC 2675, *IPv6 Jumbo grams*
- RFC 2767, *Dual Stack Hosts using the "Bump-In-the-Stack" Technique (BIS)*
- RFC 2878, *PPP Bridging Control Protocol*
- RFC 2893, *Transition Mechanisms for IPv6 Hosts and Routers*
- Internet draft draft-ietf-dhc-dhcpv6-16.txt, *Dynamic Host Configuration Protocol for IPv6* (expires May 2001)
- Internet draft draft-kato-bgp-ipv6-link-local-00.txt, *BGP4+ Peering Using IPv6 Link-local Address* (expires April 2002)
- Internet draft draft-ietf-idr-flow-spec-00.txt, *Dissemination of Flow Specification Rules*

To access Internet Requests for Comments (RFCs) and drafts, see <http://www.ietf.org>.

CHAPTER 3

Complete Routing and Routing Protocol Configuration Statements

- [\[edit logical-systems\] Hierarchy Level on page 21](#)
- [\[edit protocols\] Hierarchy Level on page 22](#)
- [\[edit routing-instances\] Hierarchy Level on page 39](#)
- [\[edit routing-options\] Hierarchy Level on page 44](#)

[\[edit logical-systems\] Hierarchy Level](#)

The following lists the statements that can be included at the [\[edit logical-systems\]](#) hierarchy level and are also documented in this manual.

```
logical-systems {  
  logical-system-name {  
    protocols {  
      bgp {  
        bgp-configuration;  
      }  
      isis {  
        isis-configuration;  
      }  
      ospf {  
        ospf-configuration;  
      }  
      ospf3 {  
        ospf3-configuration;  
      }  
      rip {  
        rip-configuration;  
      }  
      ripng {  
        ripng-configuration;  
      }  
      router-advertisement {  
        router-advertisement-configuration;  
      }  
      router-discovery {  
        router-discovery-configuration;  
      }  
    }  
  }  
}
```

```
    }
    routing-instances {
        routing-instance-name {
            routing-instance-configuration;
        }
    }
    routing-options {
        routing-option-configuration;
    }
}
```

[edit protocols] Hierarchy Level

The following statements can also be included at the **[edit logical-systems *logical-system-name*]** hierarchy level.

```
protocols {
    BGP Global    bgp {
        accept-remote-nexthop;
        advertise-external <conditional>;
        advertise-inactive;
        (advertise-peer-as | noadvertise-peer-as);
        authentication-algorithm algorithm;
        authentication-key key;
        authentication-key-chain key-chain;
        bfd-liveness-detection{
            authentication {
                algorithm algorithm-name;
                key-chain key-chain-name;
                loose-check;
            }
            detection-time {
                threshold milliseconds;
            }
            holddown-interval milliseconds;
            minimum-interval milliseconds;
            minimum-receive-interval milliseconds;
            no-adaptation;
            session-mode (automatic | multihop | single-hop);
            transmit-interval {
                threshold milliseconds;
                minimum-interval milliseconds;
            }
            multiplier number;
            version (1 | automatic);
        }
        cluster cluster-identifier;
        damping;
        description text-description;
        disable;
        export [ policy-names ];
        family family-name{
            ... family-configuration ...
        }
        graceful-restart {
```

```

    disable;
    restart-time seconds;
    stale-routes-time seconds;
}
group group-name {
... group-configuration ...
}
hold-time seconds;
idle-after-switch-over (seconds | forever);
import [ policy-names ];
include-mp-next-hop;
ipsec-sa ipsec-sa;
keep (all | none);
local-address address;
local-as autonomous-system <private>;
local-interface interface-name;
local-preference local-preference;
log-updown;
metric-out (metric | igp (delay-med-update | <metric-offset>) | minimum-igp
    <metric-offset>);
mtu-discovery;
multihop {
    no-nexthop-change;
    ttl tll-value;
}
no-aggregator-id;
no-client-reflect;
outbound-route-filter{
    bgp-orf-cisco-mode;
    prefix-based {
        accept {
            (inet | inet6);
        }
    }
}
}
out-delay seconds;
passive;
path-selection {
    (cisco-non-deterministic | always-compare-med | external-router-id);
    med-plus-igp {
        igp-multiplier number;
        med-multiplier number;
    }
}
}
peer-as autonomous-system;
preference preference;
remove-private;
tcp-mss segment-size;
traceoptions {
    file filename <files number> <size size> <world-readable | no-world-readable>;
    flag flag <flag-modifier> <disable>;
}
vpn-apply-export;
}

```

```
BGP Family    family {
                (inet | inet-vpn | inet6-vpn | iso-vpn) {
                (any | flow | labeled-unicast | multicast | unicast) {
                    accepted-prefix-limit {
                        maximum number;
                        teardown <percentage> <idle-timeout (forever | minutes)>;
                    }
                    add-path {
                        send {
                            path-count number;
                            prefix-policy [ policy-names ];
                        }
                        receive;
                    }
                    <loops number>;
                    prefix-limit {
                        maximum number;
                        teardown <percentage> <idle-timeout (forever | minutes)>;
                    }
                    rib-group group-name;
                }
                flow{
                    no-validate policy-name;
                }
                labeled-unicast {
                    accepted-prefix-limit {
                        maximum number;
                        teardown <percentage> <idle-timeout (forever | minutes)>;
                    }
                    aggregate-label {
                        community community-name;
                    }
                    explicit-null {
                        connected-only;
                    }
                    prefix-limit {
                        maximum number;
                        teardown <percentage> <idle-timeout (forever | minutes)>;
                    }
                    resolve-vpn;
                    rib inet.3;
                    rib-group group-name;
                }
            }
            route-target {
                accepted-prefix-limit {
                    maximum number;
                    teardown <percentage> <idle-timeout (forever | minutes)>;
                }
                advertise-default;
                external-paths number;
                prefix-limit {
                    maximum number;
                    teardown <percentage> <idle-timeout (forever | minutes)>;
                }
            }
        }
```

```

(inet-mdt | inet-mvpn | inet6-mvpn | l2-vpn) {
  signaling {
    accepted-prefix-limit {
      maximum number;
      teardown <percentage> <idle-timeout (forever | minutes)>;
    }
    <loops number>;
    prefix-limit {
      maximum number;
      teardown <percentage> <idle-timeout (forever | minutes)>;
    }
    rib-group group-name
  }
}

```

BGP Group

```

group group-name {
  accept-remote-nexthop;
  advertise-external <conditional>;
  advertise-inactive;
  advertise-peer-as;
  allow ([ network/mask-length ] | all);
  as-override;
  authentication-algorithm algorithm;
  authentication-key key;
  authentication-key-chain key-chain;
  bfd-liveness-detection {
    authentication {
      algorithm algorithm-name;
      key-chain key-chain-name;
      loose-check;
    }
    detection-time {
      threshold milliseconds;
    }
    holddown-interval milliseconds;
    minimum-interval milliseconds;
    minimum-receive-interval milliseconds;
    multiplier number;
    no-adaptation;
    session-mode (automatic | multihop | single-hop);
    transmit-interval {
      threshold milliseconds;
      minimum-interval milliseconds;
    }
    version (1 | automatic);
  }
  cluster cluster-identifier;
  damping;
  description text-description;
  export [ policy-names ];
  family {
    (inet | inet6 | inet-vpn | inet6-vpn | iso-vpn) {
      (any | flow | labeled-unicast | multicast | unicast) {
        accepted-prefix-limit {
          maximum number;
          teardown <percentage> <idle-timeout (forever | minutes)>;

```

```
    }
    add-path {
        send {
            path-count number;
            prefix-policy [ policy-names ];
        }
        receive;
    }
    <loops number>;
    prefix-limit {
        maximum number;
        teardown <percentage> <idle-timeout (forever | minutes)>;
    }
    rib-group group-name;
}
flow {
    no-validate policy-name;
}
labeled-unicast {
    accepted-prefix-limit {
        maximum number;
        teardown <percentage> <idle-timeout (forever | minutes)>;
    }
    aggregate-label {
        community community-name;
    }
    explicit-null {
        connected-only;
    }
    prefix-limit {
        maximum number;
        teardown <percentage> <idle-timeout (forever | minutes)>;
    }
    resolve-vpn;
    rib inet.3;
    rib-group group-name;
}
}
route-target {
    accepted-prefix-limit {
        maximum number;
        teardown <percentage> <idle-timeout (forever | minutes)>;
    }
    advertise-default;
    external-paths number;
    prefix-limit {
        maximum number;
        teardown <percentage> <idle-timeout (forever | minutes)>;
    }
}
}
(inet-mdt | inet-mvpn | inet6-mvpn | l2-vpn) {
    signaling {
        accepted-prefix-limit {
            maximum number;
            teardown <percentage> <idle-timeout (forever | minutes)>;
        }
    }
}
```



```

        <loops number>;
        prefix-limit {
            maximum number;
            teardown <percentage> <idle-timeout (forever | minutes)>;
        }
        rib-group group-name
    }
}
graceful-restart {
    disable;
    restart-time seconds;
    stale-routes-time seconds;
}
hold-time seconds;
idle-after-switch-over (seconds | forever);
import [ policy-names ];
include-mp-next-hop;
ipsec-sa ipsec-sa;
keep (all | none);
local-address address;
local-as autonomous-system <private>;
local-interface;
local-preference local-preference;
log-updown;
metric-out (metric | minimum-igp <offset> | igp <offset>);
mtu-discovery;
multihop <ttl-value>;
multipath {
    multiple-as;
}
no-advertise-peer-as;
no-aggregator-id;
no-client-reflect;
outbound-route-filter {
    bgp-orf-cisco-mode;
    prefix-based {
        accept {
            (inet | inet6);
        }
    }
}
}
out-delay seconds;
passive;
peer-as autonomous-system;
preference preference;
remove-private;
tcp-mss segment-size;
traceoptions {
    file filename <files number> <size size> <world-readable | no-world-readable>;
    flag flag <flag-modifier> <disable>;
}
type type;
vpn-apply-export;

```

```
BGP Neighbor      neighbor address {
                    accept-remote-nexthop;
                    advertise-external <conditional>;
                    advertise-inactive;
                    advertise-peer-as;
                    as-override;
                    authentication-algorithm algorithm;
                    authentication-key key;
                    authentication-key-chain key-chain;
                    bfd-liveness-detection {
                        authentication {
                            algorithm algorithm-name;
                            key-chain key-chain-name;
                            loose-check;
                        }
                        detection-time {
                            threshold milliseconds;
                        }
                        minimum-interval milliseconds;
                        minimum-receive-interval milliseconds;
                        multiplier number;
                        no-adaptation;
                        session-mode (automatic | multihop | single-hop);
                        transmit-interval {
                            threshold milliseconds;
                            minimum-interval milliseconds;
                        }
                        version (1 | automatic);
                    }
                    cluster cluster-identifier;
                    damping;
                    description text-description;
                    export [ policy-names ];
                    family{
                        (inet | inet6 | inet-vpn | inet6-vpn | iso-vpn) {
                            (any | flow | labeled-unicast | multicast | unicast) {
                                accepted-prefix-limit {
                                    maximum number;
                                    teardown <percentage> <idle-timeout (forever | minutes)>;
                                }
                                add-path {
                                    send {
                                        path-count number;
                                        prefix-policy [ policy-names ];
                                    }
                                    receive;
                                }
                                <loops number>;
                                prefix-limit {
                                    maximum number;
                                    teardown <percentage> <idle-timeout (forever | minutes)>;
                                }
                                rib-group group-name;
                            }
                        }
                        flow {
                            no-validate policy-name;
                        }
                    }
                }
```

```

}
labeled-unicast {
  accepted-prefix-limit {
    maximum number;
    teardown <percentage> <idle-timeout (forever | minutes)>;
  }
  aggregate-label {
    community community-name;
  }
  explicit-null {
    connected-only;
  }
  prefix-limit {
    maximum number;
    teardown <percentage> <idle-timeout (forever | minutes)>;
  }
  resolve-vpn;
  rib inet.3;
  rib-group group-name;
}
}
route-target {
  accepted-prefix-limit {
    maximum number;
    teardown <percentage> <idle-timeout (forever | minutes)>;
  }
  advertise-default;
  external-paths number;
  prefix-limit {
    maximum number;
    teardown <percentage> <idle-timeout (forever | minutes)>;
  }
}
}
(inet-mdt | inet-mvpn | inet6-mvpn | l2-vpn) {
  signaling {
    accepted-prefix-limit {
      maximum number;
      teardown <percentage> <idle-timeout (forever | minutes)>;
    }
    <loops number>;
    prefix-limit {
      maximum number;
      teardown <percentage> <idle-timeout (forever | minutes)>;
    }
    rib-group group-name
  }
}
}
graceful-restart {
  disable;
  restart-time seconds;
  stale-routes-time seconds;
}
hold-time seconds;
idle-after-switch-over (seconds | forever);
import [ policy-names ];

```

```
include-mp-next-hop;
ipsec-sa ipsec-sa;
keep (all | none);
local-address address;
local-as autonomous-system <private>;
local-interface interface-name;
local-preference local-preference;
log-updown;
metric-out (metric | minimum-igp <offset> | igp <offset>);
mtu-discovery;
multihop <ttl-value>;
multipath {
    multiple-as;
}
no-advertise-peer-as;
no-aggregator-id;
no-client-reflect;
out-delay seconds;
outbound-route-filter {
    bgp-orf-cisco-mode;
    prefix-based {
        accept {
            (inet | inet6);
        }
    }
}
passive;
peer-as autonomous-system;
preference preference;
remove-private;
tcp-mss segment-size;
traceoptions {
    file filename <files number> <size size> <world-readable | no-world-readable>;
    flag flag <flag-modifier> <disable>;
}
vpn-apply-export;
```

```
IS-IS isis {
    clns-routing;
    disable;
    export [ policy-names ];
    graceful-restart {
        disable;
        helper-disable;
        restart-duration seconds;
    }
    ignore-attached-bit;
    interface (all | interface-name) {
        bfd-liveness-detection {
            authentication {
                algorithm algorithm-name;
                key-chain key-chain-name;
                loose-check;
            }
            detection-time {
                threshold milliseconds;
            }
        }
    }
}
```

```

    }
    minimum-interval milliseconds;
    minimum-receive-interval milliseconds;
    transmit-interval {
        threshold milliseconds;
        minimum-interval milliseconds;
    }
    multiplier number;
}
checksum;
csnp-interval (seconds | disable);
disable;
hello-padding (adaptive | loose | strict);
ldp-synchronization {
    disable;
    hold-time seconds;
}
level level-number {
    disable;
    hello-authentication-key key;
    hello-authentication-key-chain key-chain-name;
    hello-authentication-type authentication;
    hello-interval seconds;
    hold-time seconds;
    ipv4-multicast-metric number;
    ipv6-multicast-metric number;
    ipv6-unicast-metric number;
    metric metric;
    passive;
    priority number;
    te-metric metric;
}
link-protection;
lsp-interval milliseconds;
mesh-group (value | blocked);
no-adjacency-holddown;
no-eligible-backup;
no-ipv4-multicast;
no-ipv6-multicast;
no-ipv6-unicast;
no-unicast-topology;
node-link-protection;
passive;
point-to-point;
}
label-switched-path name level level metric metric;
level level-number {
    authentication-key key;
    authentication-key-chain key-chain-name;
    authentication-type authentication;
    external-preference preference;
    ipv6-multicast-metric number;
    no-csnp-authentication;
    no-hello-authentication;
    no-psnp-authentication;
    preference preference;

```

```
    prefix-export-limit number;  
    wide-metrics-only;  
}  
loose-authentication-check;  
lsp-lifetime seconds;  
max-areas number;  
no-adjacency-holddown;  
no-authentication-check;  
no-ipv4-routing;  
no-ipv6-routing;  
overload {  
    advertise-high-metrics;  
    timeout seconds;  
}  
reference-bandwidth reference-bandwidth;  
rib-group {  
    inet group--name;  
    inet6 group--name;  
}  
spf-options {  
    delay milliseconds;  
    holddown milliseconds;  
    rapid-runs number;  
}  
topologies {  
    ipv4-multicast;  
    ipv6-multicast;  
    ipv6-unicast;  
}  
traceoptions {  
    file filename <files number> <size size> <world-readable | no-world-readable>;  
    flag flag <flag-modifier> <disable>;  
}  
traffic-engineering {  
    disable;  
    family inet {  
        shortcuts <ignore-lsp-metrics> {  
            multicast-rpf-routes;  
        }  
    }  
    family inet6 {  
        shortcuts;  
    }  
}  
}
```

```
OSPF ospf {  
    disable;  
    export [ policy-names ];  
    external-preference preference;  
    graceful-restart {  
        disable;  
        helper-disable;  
        notify-duration seconds;  
        restart-duration seconds;  
    }  
}
```

```

import [ policy-names ];
no-nssa-abr;
overload {
    timeout seconds;
}
preference preference;
reference-bandwidth reference-bandwidth;
rib-group group-name;
sham-link {
    local address;
}
spf-options {
    delay milliseconds;
    holddown milliseconds;
    rapid-runs milliseconds;
}
traffic-engineering {
    accept-unnumbered-interfaces;
    multicast-rpf-routes;
    no-topology;
    shortcuts {
        ignore-lsp-metrics;
        lsp-metric-into-summary;
    }
}
traceoptions {
    file filename <files number> <size size> <world-readable | no-world-readable>;
    flag flag <flag-modifier> <disable>;
}
area area-id {
    area-range network/mask-length <restrict> <exact> <override-metric metric>;
    interface interface-name {
        demand-circuit;
        disable;
        bfd-liveness-detection {
            authentication {
                algorithm algorithm-name;
                key-chain key-chain-name;
                loose-check;
            }
            detection-time {
                threshold milliseconds;
            }
            minimum-interval milliseconds;
            minimum-receive-interval milliseconds;
            multiplier number;
            no-adaptation;
            transmit-interval {
                threshold milliseconds;
                minimum-interval milliseconds;
            }
            version (1 | automatic);
        }
        ipsec-sa name;
    }
    no-interface-state-traps;
}

```

```
authentication {
  md5 key-id {
    key [ key-values ];
  }
  simple-password key-id;
}
dead-interval seconds;
hello-interval seconds;
interface-type type;
ldp-synchronization {
  disable;
  hold-time seconds;
}
metric metric;
neighbor address <eligible>;
network-summary-export [ policy-names ];
network-summary-import [ policy-names ];
passive;
poll-interval seconds;
priority number;
retransmit-interval seconds;
te-metric metric;
transit-delay seconds;
}
label-switched-path name metric metric;
nssa {
  area-range network/mask-length <restrict> <exact> <override-metric metric>;
  default-lsa {
    default-metric metric;
    metric-type type;
    type-7;
  }
  (no-summaries | summaries);
}
peer-interface interface-name {
  disable;
  dead-interval seconds;
  hello-interval seconds;
  retransmit-interval seconds;
  transit-delay seconds;
}
sham-link-remote address {
  ipsec-sa name;
}
demand-circuit;
metric metric;
}
stub <default-metric metric> <(no-summaries | summaries)>;
virtual-link neighbor-id router-id transit-area area-id {
  disable;
  ipsec-sa name;
}
authentication {
  md5 key-id;
  simple-password key-id;
}
```



```

dead-interval seconds;
hello-interval seconds;
retransmit-interval seconds;
transit-delay seconds;

OSPFv3  ospf3 {
        disable;
        export [ policy-names ];
        external-preference preference;
        import [ policy-names ];
        overload {
            timeout seconds;
        }
        preference preference;
        reference-bandwidth reference-bandwidth;
        rib-group group-name;
        spf-options {
            delay milliseconds;
            holddown milliseconds;
            rapid-runs number;
        }
        traceoptions {
            file filename <files number> <size size> <world-readable | no-world-readable>;
            flag flag <flag-modifier> <disable>;
        }
        area area-id {
            area-range network/mask-length <restrict> <exact> <override-metric metric>;
            interface interface-name {
                disable;
                dead-interval seconds;
                hello-interval seconds;
                ipsec-sa name;
                metric metric;
                neighbor address <eligible>;
                passive;
                priority number;
                retransmit-interval seconds;
                transit-delay seconds;
            }
            inter-area-prefix-export [ policy-names ];
            inter-area-prefix-import [ policy-names ];
            nssa {
                area-range network/mask-length <restrict> <exact> <override-metric metric>;
                default-lsa {
                    default-metric metric;
                    metric-type type;
                    type-7;
                }
                (no-summaries | summaries);
            }
            stub <default-metric metric> <(no-summaries | summaries)>;
            virtual-link neighbor-id router-id transit-area area-id {
                disable;
                dead-interval seconds;
                hello-interval seconds;
                ipsec-sa name;
            }
        }
    }

```

```
        retransmit-interval seconds;  
        transit-delay seconds;  
    }  
}  
}  
  
RIP  rip {  
    any-sender;  
    authentication-key password;  
    authentication-type type;  
    (check-zero | no-check-zero);  
    graceful-restart {  
        disable;  
        restart-time seconds;  
    }  
    holddown seconds;  
    import [ policy-names ];  
    message-size number;  
    metric-in metric;  
    receive receive-options;  
    rib-group group-name;  
    route-timeout seconds;  
    send send-options;  
    update-interval seconds;  
    traceoptions {  
        file filename <files number> <size size> <world-readable | no-world-readable>;  
        flag flag <flag-modifier> <disable>;  
    }  
    group group-name {  
        bfd-liveness-detection {  
            authentication {  
                algorithm algorithm-name;  
                key-chain key-chain-name;  
                loose-check;  
            }  
            detection-time {  
                threshold milliseconds;  
            }  
            minimum-interval milliseconds;  
            minimum-receive-interval milliseconds;  
            multiplier number;  
            no-adaptation;  
            transmit-interval {  
                threshold milliseconds;  
                minimum-interval milliseconds;  
            }  
            version (0 | 1 | automatic);  
        }  
        export [ policy-names ];  
        metric-out metric;  
        preference preference;  
        route-timeout seconds;  
        update-interval seconds;  
        neighbor neighbor-name {  
            authentication-key password;  
            authentication-type type;  
        }  
    }  
}
```

```

bfd-liveness-detection {
  authentication {
    algorithm algorithm-name;
    key-chain key-chain-name;
    loose-check;
  }
  detection-time {
    threshold milliseconds;
  }
  minimum-interval milliseconds;
  minimum-receive-interval milliseconds;
  transmit-interval {
    threshold milliseconds;
    minimum-interval milliseconds;
  }
  multiplier number;
  version (1 | automatic);
}
(check-zero | no-check-zero);
import [ policy-names ];
message-size number;
metric-in metric;
receive receive-options;
route-timeout seconds;
send send-options;
update-interval seconds;
}
}
}

```

```

RIPng ripng {
  graceful-restart {
    disable;
    restart-time seconds;
  }
  holddown seconds;
  import [ policy-names ];
  metric-in metric;
  receive <none>;
  route-timeout seconds;
  send <none>;
  update-interval seconds;
  traceoptions {
    file filename <files number> <size size> <world-readable | no-world-readable>;
    flag flag <flag-modifier> <disable>;
  }
  group group-name {
    export [ policy-names ];
    metric-out metric;
    preference number;
    route-timeout seconds;
    update-interval seconds;
    neighbor neighbor-name {
      import [ policy-names ];
      metric-in metric;
      receive <none>;
    }
  }
}

```

	<pre> route-timeout <i>seconds</i>; send <none>; update-interval <i>seconds</i>; } }</pre>
Router Advertisement	<pre>router-advertisement { interface <i>interface-name</i> { current-hop-limit <i>number</i>; default-lifetime <i>seconds</i>; (link-mtu no-link-mtu); (managed-configuration no-managed-configuration); max-advertisement-interval <i>seconds</i>; min-advertisement-interval <i>seconds</i>; (other-stateful-configuration no-other-stateful-configuration); prefix <i>prefix</i> { (autonomous no-autonomous); (on-link no-on-link); preferred-lifetime <i>seconds</i>; valid-lifetime <i>seconds</i>; } reachable-time <i>milliseconds</i>; retransmit-timer <i>milliseconds</i>; traceoptions { file <i>filename</i> <files <i>number</i>> <size <i>size</i>> <world-readable no-world-readable>; flag <i>flag</i> <detail> <disable>; } } }</pre>
Router Discovery	<pre>router-discovery { disable; interface <i>interface-name</i> { min-advertisement-interval <i>seconds</i>; max-advertisement-interval <i>seconds</i>; lifetime <i>seconds</i>; } address <i>address</i> { (advertise ignore); (broadcast multicast); (priority <i>number</i> ineligible); } }</pre>
Secure Neighbor Discovery	<pre>neighbor-discovery { secure { security-level { (default secure-messages-only); } cryptographic-address { key-length <i>number</i>; key-pair <i>pathname</i>; } timestamp {</pre>

```

        clock-drift number;
        known-peer-window seconds;
        new-peer-window seconds;
    }
    traceoptions {
        file filename <files number> <match regular-expression> <size size> <world-readable |
            no-world-readable>;
        flag flag;
        no-remote-trace;
    }
}
}

```

[\[edit routing-instances\] Hierarchy Level](#)

```

routing-instances {
    routing-instance-name {
        bridge-domains bridge-domain-name {
            domain-type bridge;
            <vlan-id (all | none | number)>;
            <vlan-tags outer number inner number>;
            <routing-interface routing-interface-name>;
            interface interface-name;
            bridge-options {
                interface-mac-limit limit;
                mac-statistics;
                mac-table-size limit;
                no-mac-learning;
                static-mac mac-address;
            }
        }
        description text;
        forwarding-options;
        interface interface-name;
        instance-type (forwarding | layer2-control | l2vpn | no-forwarding | virtual-router |
            virtual-switch | vpls | vrf);
        no-vrf-advertise;
        no-vrf-propagate-ttl;
        route-distinguisher (as-number:number | ip-address:number);
        vrf-import [ policy-names ];
        vrf-export [ policy-names ];
        vrf-propagate-ttl;
        vrf-table-label;
        vrf-target {
            export community-name;
            import community-name;
        }
        protocols {
            bgp {
                bgp-configuration;
            }
            isis {
                isis-configuration;
            }
            l2vpn {

```

```
    l2vpn-configuration;
}
ldp {
    ldp-configuration;
}
msdp {
    msdp-configuration;
}
ospf {
    domain-id domain-id;
    domain-vpn-tag number;
    route-type-community (vendor | iana);
    ospf-configuration;
}
ospf 3 {
    domain-id domain-id;
    domain-vpn-tag number;
    route-type-community (vendor | iana);
    ospf3-configuration;
}
pim {
    pim-configuration;
}
rip {
    rip-configuration;
}
vpls {
    vpls-configuration;
}
}
routing-options {
    aggregate {
        defaults {
            ... aggregate-options ...
        }
        route destination-prefix {
            policy policy-name;
            ... aggregate-options ...
        }
    }
}
auto-export {
    (disable | enable);
    family {
        inet {
            flow {
                (disable | enable);
                rib-group rib-group;
            }
            multicast {
                (disable | enable);
                rib-group rib-group;
            }
            unicast {
                (disable | enable);
                rib-group rib-group;
            }
        }
    }
}
```

```

    }
  }
  traceoptions {
    file filename <files number> <size size> <world-readable | no-world-readable>;
    flag flag <flag-modifier> <disable>;
  }
}
autonomous-system autonomous-system <loops number> {
  independent-domain <no-attrset>;
}
confederation confederation-autonomous-systems
members autonomous-system;
dynamic-tunnels tunnel-name {
  destination-prefix prefix;
  source-address address;
  tunnel-type type-of-tunnel;
}
fate-sharing {
  group group-name;
  cost value;
  from address {
    to address;
  }
  flow {
    route name {
      match {
        match-conditions;
      }
      then {
        actions;
      }
    }
  }
  validation {
    traceoptions {
      file filename <files number> <size size> <world-readable | no-world-readable>;
      flag flag <flag-modifier> <disable>;
    }
  }
}
generate {
  defaults {
    generate-options;
  }
  route destination-prefix {
    policy policy-name;
    generate-options;
  }
}
instance-export [ policy-names ];
instance-import [ policy-names ];
interface-routes {
  family (inet | inet6) {
    export {
      lan;
      point-to-point;
    }
  }
}

```

```
    rib-group group-name;
}
martians {
    destination-prefix match-type <allow>;
}
maximum-paths path-limit <log-only | threshold value log-interval seconds>;
maximum-prefixes prefix-limit <log-only | threshold value log-interval seconds>;
multicast {
    forwarding-cache {
        threshold (suppress | reuse) value value;
    }
    interface interface-name {
        enable;
    }
    scope scope-name {
        interface interface-name;
        prefix destination-prefix;
    }
    scope-policy policy-name;
    ssm-groups {
        addresses;
    }
}
options {
    syslog (level level | upto level);
}
resolution {
    rib routing-table-name {
        import [ policy-names ];
        resolution-ribs [ routing-table-names ];
    }
}
rib routing-table-name {
    aggregate {
        defaults {
            ... aggregate-options ...
        }
        route destination-prefix {
            policy policy-name;
            ... aggregate-options ...
        }
    }
    filter {
        input filter-name;
    }
    generate {
        defaults {
            generate-options;
        }
        route destination-prefix {
            policy policy-name;
            generate-options;
        }
    }
}
martians {
    destination-prefix match-type <allow>;
```



```
}
static {
  defaults {
    static-options;
  }
  passive group-name;
  route destination-prefix {
    lsp-next-hop {
      metric metric;
      preference preference;
    }
    next-hop;
    p2mp-lsp-next-hop {
      metric metric;
      preference preference;
    }
    qualified-next-hop address {
      interface interface-name;
      metric metric;
      preference preference;
    }
    static-options;
  }
}
}
passive {
  group-name {
    import-policy [ policy-names ];
    import-rib [ group-names ];
    export-rib group-name;
  }
}
route-distinguisher-id address;
route-record;
router-id address;
static {
  defaults {
    static-options;
  }
  rib-group group-name;
  route destination-prefix {
    bfd-liveness-detection {
      authentication {
        algorithm algorithm-name;
        key-chain key-chain-name;
        loose-check;
      }
      detection-time {
        threshold milliseconds;
      }
      local-address ip-address;
      minimum-interval milliseconds;
      minimum-receive-interval milliseconds;
      minimum-receive-ttl milliseconds;
      transmit-interval {
        threshold milliseconds;
      }
    }
  }
}
```

```

        minimum-interval milliseconds;
    }
    multiplier number;
    version (1 | automatic);
}
lsp-next-hop {
    metric metric;
    preference preference;
}
next-hop;
qualified-next-hop address {
    interface interface-name;
    metric metric;
    preference preference;
}
static-options;
}
}
traceoptions {
    file filename <files number> <size size> <world-readable | no-world-readable>;
    flag flag <flag-modifier> <disable>;
}
}
}
}
}

```

The following statements can also be included at the [edit logical-systems *logical-system-name*] hierarchy level.



NOTE: The virtual-switch instance type is not supported at the [edit logical-systems *logical-system-name*] hierarchy level. For more detailed information about configuring a virtual switch on MX Series routers, see the [Junos OS Layer 2 Configuration Guide](#).

[edit routing-options] Hierarchy Level

The following statements can also be included at the [edit logical-systems *logical-system-name*] hierarchy level.

```

routing-options {
    aggregate {
        defaults {
            ... aggregate-options ...
        }
        route destination-prefix {
            policy policy-name;
            ... aggregate-options ...
        }
    }
    auto-export {
        (disable | enable);
    }
}

```

```

family {
  inet {
    flow {
      (disable | enable);
      rib-group rib-group;
    }
    multicast {
      (disable | enable);
      rib-group rib-group;
    }
    unicast {
      (disable | enable);
      rib-group rib-group;
    }
  }
}
traceoptions {
  file filename <files number> <size size> <world-readable | no-world-readable>;
  flag flag <flag-modifier> <disable>;
}
}
autonomous-system autonomous-system <loops number>;
confederation confederation-autonomous-system members autonomous-system;
dynamic-tunnels tunnel-name {
  destination-prefix prefix;
  source-address address;
  tunnel-type tunnel-type;
}
fate-sharing {
  group group-name;
  cost value;
  from address {
    to address;
  }
}
flow {
  route name {
    match {
      match-conditions;
    }
    then {
      actions;
    }
  }
}
validation {
  traceoptions {
    file filename <files number> <size size> <world-readable | no-world-readable>;
    flag flag <flag-modifier> <disable>;
  }
}
forwarding-table {
  export [ policy-names ];
  (indirect-next-hop | no-indirect-next-hop);
  unicast-reverse-path (active-paths | feasible-paths);
}
generate {

```

```
defaults {
    generate-options;
}
route destination-prefix {
    policy policy-name;
    generate-options;
}
}
graceful-restart {
    disable;
    restart-duration seconds;
}
instance-export [ policy-names ];
instance-import [ policy-names ];
interface-routes {
    family (inet | inet6) {
        export {
            lan;
            point-to-point;
        }
    }
    rib-group group-name;
}
martians {
    destination-prefix match-type <allow>;
}
maximum-paths path-limit <log-only | threshold value log-interval seconds>;
maximum-prefixes prefix-limit <log-only | threshold value log-interval seconds>;
multicast {
    forwarding-cache {
        threshold (suppress | reuse) value value;
    }
    interface interface-name {
        enable;
    }
    scope scope-name {
        interface interface-name;
        prefix destination-prefix;
    }
    scope-policy policy-name;
    ssm-groups {
        address;
    }
}
options {
    syslog (level level | upto level);
}
ppm {
    delegate-processing;
}
resolution {
    rib routing-table-name {
        import [ policy-names ];
        resolution-ribs [ routing-table-names ];
    }
}
}
```

```

rib routing-table-name {
  aggregate {
    defaults {
      ... aggregate-options ...
    }
    rib-group group-name;
    route destination-prefix {
      policy policy-name;
      ... aggregate-options ...
    }
  }
  filter {
    input filter-name;
  }
  generate {
    defaults {
      generate-options;
    }
    route destination-prefix {
      policy policy-name;
      generate-options;
    }
  }
  martians {
    destination-prefix match-type <allow>;
  }
  static {
    defaults {
      static-options;
    }
    rib-group group-name;
    route destination-prefix {
      lsp-next-hop {
        metric metric;
        preference preference;
      }
      next-hop;
      qualified-next-hop address {
        interface interface-name;
        metric metric;
        preference preference;
      }
      static-options;
    }
  }
}
rib-groups {
  group-name {
    import-policy [ policy-names ];
    import-rib [ group-names ];
    export-rib group-name;
  }
}
route-distinguisher-id address;
route-record;
router-id address;

```

```
static {
  defaults {
    static-options;
  }
  rib-group group-name;
  route destination-prefix {
    bfd-liveness-detection {
      authentication {
        algorithm algorithm-name;
        key-chain key-chain-name;
        loose-check;
      }
      detection-time {
        threshold milliseconds;
      }
      holddown-interval milliseconds;
      local-address ip-address;
      minimum-interval milliseconds;
      minimum-receive-interval milliseconds;
      minimum-receive-ttl number;
      multiplier number;
      neighbor address;
      no-adaptation;
      transmit-interval {
        threshold milliseconds;
        minimum-interval milliseconds;
      }
      version (1 | automatic);
    }
    lsp-next-hop {
      metric metric;
      preference preference;
    }
    next-hop;
    p2mp-lsp-next-hop {
      metric metric;
      preference preference;
    }
    qualified-next-hop (address | interface-name) {
      interface interface-name;
      metric metric;
      preference preference;
    }
    source-routing {
      (ip | ipv6);
    }
    static-options;
  }
}
topologies {
  (inet | inet6) {
    topology name;
  }
}
traceoptions {
  file filename <files number> <size size> <world-readable | no-world-readable>;
```

```
        flag flag <flag-modifier> <disable>;  
    }  
}  
}
```


PART 3

Troubleshooting

- [Routing Protocol Process Memory FAQ on page 53](#)

CHAPTER 4

Routing Protocol Process Memory FAQ

- [Routing Protocol Process Memory FAQ Overview on page 53](#)
- [Routing Protocol Process Memory FAQs on page 54](#)

Routing Protocol Process Memory FAQ Overview

The Juniper Networks Junos operating system (Junos OS) is based on the FreeBSD Unix operating system. The open source software is modified and hardened to operate in the device's specialized environment. For example, some executables have been deleted while other utilities have been de-emphasized. Additionally, certain software processes have been added to enhance the routing functionality. The result of this transformation is the kernel, the heart of the Junos OS software.

The kernel is responsible for generating multiple processes that perform the actual functions of the device. Each process operates in its own protected memory space, providing isolation between the processes and resiliency in the event of a process failure. This is important in a core routing platform because a single process failure does not cause the entire device to cease functioning.

Some of the common software processes include the routing protocol process (rpd) that controls the device's protocols, the device control process (dcd) that controls the device's interfaces, the management process (mgd) that controls user access to the device, the chassis process (chassisd) that controls the device's properties itself, and the Packet Forwarding Engine process (pfed) that controls the communication between the device's Packet Forwarding Engine and the Routing Engine. Besides the above processes, there are other specialized processes that support additional functionality, such as the Simple Network Management Protocol (SNMP), Virtual Router Redundancy Protocol (VRRP), and Class of Service (CoS).

The routing protocol process is a software process within the Routing Engine software that controls the routing protocols that run on the device. Its functionality includes all protocol messages, routing table updates, and implementation of routing policies.

The routing protocol process starts all configured routing protocols and handles all routing messages. It maintains one or more routing tables, which consolidate the routing information learned from all routing protocols. From this routing information, the routing protocol process determines the active routes to network destinations and installs these routes into the Routing Engine's forwarding table. Finally, it implements the routing policy, which allows you to control the routing information that is transferred between the routing

protocols and the routing table. Using the routing policy, you can filter and limit the transfer of information as well as set properties associated with specific routes.

Related Documentation • [Routing Protocol Process Memory FAQs on page 54](#)

Routing Protocol Process Memory FAQs

The following sections present the most frequently asked questions and answers related to the routing protocol process memory utilization, operation, interpretation of related command outputs, and troubleshooting the software process.

Routing Protocol Process Memory Utilization FAQs

This section presents frequently asked questions and answers related to the memory usage of the routing protocol process.

Why does the routing protocol process use excessive memory?

The routing protocol process uses hundreds of megabytes of RAM in the Routing Engine to store information needed for the operation of routing and related protocols, such as BGP, OSPF, ISIS, RSVP, LDP, and MPLS. Such huge consumption of memory is common for the process, as the information it stores includes routes, next hops, interfaces, routing policies, labels, and label-switched paths (LSPs). Because access to the RAM memory is much faster than access to the hard disk, most of the routing protocol process information is stored in the RAM memory instead of using the hard disk space. This ensures that the performance of the routing protocol process is maximized.

How can I check the amount of memory the routing protocol process is using?

You can check the routing protocol process memory usage by entering the **show system processes** and the **show task memory** Junos OS command-line interface (CLI) operational mode commands.

The **show system processes** command displays information about software processes that are running on the device. You can check the routing protocol process memory usage by using the **show system processes** command with the **extensive** option.

The **show task memory** command displays a report generated by the routing protocol process on the memory utilization for routing protocol tasks on the Routing Engine. Although the report generated by the routing protocol process is on its own memory usage, it does not display all the memory used by the process. The value reported by the routing protocol process does not account for the memory used for the **TEXT** and **STACK** segments, or the memory used by the process's internal memory manager. The **show task memory** command also does not include the memory which has been deactivated by the routing protocol process, although some or all of that deactivated memory has not actually been freed by the kernel.

For more information about checking the routing protocol process memory usage, see [Check Routing Protocol Process \(rpd\) Memory Usage](#) in the *Junos OS Baseline Network Operations Guide*.

For more information about the `show system processes` command and the `show task memory` command, see the [Junos OS System Basics and Services Command Reference](#).

I just deleted many routes from the routing protocol process. Why is the routing protocol process still using so much memory?

The **show system processes extensive** command displays a **RES** value measured in kilobytes. This value represents the amount of process memory resident in the physical memory. This is also known as RSS or Resident Set Size. Any amount of memory deactivated by the process might still be considered part of the **RES** value. Generally, the kernel defers the actual freeing of deactivated memory until there is a memory shortage. This can lead to large discrepancies between the values reported by the routing protocol process and the kernel, even after the routing protocol process has deactivated a large amount of memory.

Interpreting Routing Protocol Process-Related Command Outputs FAQs

This section presents frequently asked questions and answers about the routing protocol process-related Junos OS CLI command outputs that are used to display the memory usage of the routing protocol process.

How do I interpret memory numbers displayed in the show system processes extensive command output?

The **show system processes extensive** command displays exhaustive system process information about software processes that are running on the device. This command is equivalent to the UNIX **top** command. However, the UNIX **top** command shows real-time memory usage, with the memory values constantly changing, while the **show system processes extensive** command provides a snapshot of memory usage in a given moment.

To check overall CPU and memory usage, enter the **show system processes extensive** command. Refer to [Table 4 on page 57](#) for information about the **show system processes extensive** command output fields.

```
user@host> show system processes extensive
last pid: 544; load averages: 0.00, 0.00, 0.00 18:30:33
37 processes: 1 running, 36 sleeping

Mem: 25M Active, 3968K Inact, 19M Wired, 184K Cache, 8346K Buf, 202M Free
Swap: 528M Total, 64K Used, 528M Free

  PID USERNAME PRI NICE SIZE RES STATE  TIME  WCPU   CPU COMMAND
    544 root      30  0  604K 768K RUN    0:00  0.00%  0.00% top
      3 root      28  0    0K 12K psleep 0:00  0.00%  0.00% vmdaemon
      4 root      28  0    0K 12K update 0:03  0.00%  0.00% update
    528 aviva     18  0  660K 948K pause  0:00  0.00%  0.00% tcsh
    204 root      18  0  300K 544K pause  0:00  0.00%  0.00% csh
    131 root      18  0  332K 532K pause  0:00  0.00%  0.00% cron
    186 root      18  0  196K 68K pause  0:00  0.00%  0.00% watchdog
     27 root      10  0  512M 16288K mfsidl 0:00  0.00%  0.00% mount_mfs
      1 root      10  0  620K 344K wait   0:00  0.00%  0.00% init
    304 root       3  0  884K 900K ttyin  0:00  0.00%  0.00% bash
    200 root       3  0  180K 540K ttyin  0:00  0.00%  0.00% getty
    203 root       3  0  180K 540K ttyin  0:00  0.00%  0.00% getty
    202 root       3  0  180K 540K ttyin  0:00  0.00%  0.00% getty
    201 root       3  0  180K 540K ttyin  0:00  0.00%  0.00% getty
    194 root       2  0 2248K 1640K select 0:11  0.00%  0.00% rpd
    205 root       2  0  964K 800K select 0:12  0.00%  0.00% tnp.chassisd
    189 root       2 -12 352K 740K select 0:03  0.00%  0.00% xntpd
    114 root       2  0  296K 612K select 0:00  0.00%  0.00% amd
```

```

188 root      2   0   780K   600K select  0:00  0.00%  0.00% dcd
527 root      2   0   176K   580K select  0:00  0.00%  0.00% rlogind
195 root      2   0   212K   552K select  0:00  0.00%  0.00% inetd
187 root      2   0   192K   532K select  0:00  0.00%  0.00% tnetd
 83 root      2   0   188K   520K select  0:00  0.00%  0.00% syslogd
538 root      2   0  1324K   516K select  0:00  0.00%  0.00% mgd
 99 daemon    2   0   176K   492K select  0:00  0.00%  0.00% portmap
163 root      2   0   572K   420K select  0:00  0.00%  0.00% nsrexecd
192 root      2   0   560K   400K select  0:10  0.00%  0.00% snmpd
191 root      2   0  1284K   376K select  0:00  0.00%  0.00% mgd
537 aviva     2   0   636K   364K select  0:00  0.00%  0.00% cli
193 root      2   0   312K   204K select  0:07  0.00%  0.00% mib2d
  5 root      2   0      0K    12K pfesel  0:00  0.00%  0.00% if_pfe
  2 root     -18   0      0K    12K psleep  0:00  0.00%  0.00% pagedaemon
  0 root     -18   0      0K      0K sched   0:00  0.00%  0.00% swapper

```

Table 4 on page 57 describes the output fields that represent the memory values for the **show system processes extensive** command. Output fields are listed in the approximate order in which they appear.

Table 4: show system processes extensive Output Fields

Field Name	Field Description
Mem	Information about physical and virtual memory allocation.
Active	Memory allocated and actively used by the process.
Inact	Memory allocated but not recently used, or memory deactivated by the processes. Inactive memory remains mapped in the address space of one or more processes and, therefore, counts toward the RSS value of those processes.
Wired	Memory that is not eligible to be swapped, usually used for in-kernel memory structure, memory physically locked by a process, or both.
Cache	Freed memory that is no longer associated with any process but still has valid contents that correspond to some file system blocks. Cache pages can be reclaimed as is when the corresponding file system blocks are accessed again. However, when the system is under memory pressure, the contents of Cache pages could be erased by the kernel and the pages reused to service any memory allocation requests.
Buf	Size of the virtual memory buffer used to hold data recently called from the disk.
Free	Free memory that is neither associated with any process nor contains any valid contents.
Swap	Information about swap memory. <ul style="list-style-type: none"> • Total—Total space on the swap device. • Used—Memory swapped to disk. • Free—Unused space available on the swap device.

The rest of the command output displays information about the memory usage of each process. The **SIZE** field indicates the size of the virtual address space, and the **RES** field indicates the amount of the process in physical memory, which is also known as RSS or Resident Set Size. For more information, see the **show system processes** command in the *Junos OS System Basics and Services Command Reference*.

What is the difference between Active and Inact memory that is displayed by the show system processes extensive command?

When the system is under memory pressure, the pageout process can free up memory from the **Inact** and, if necessary, **Active** pools after first preserving the contents of those pages on the swap device or backing file systems if necessary. When the pageout process runs, it scans memory to see which pages are good candidates to be unmapped and freed up. Thus, the distinction between **Active** and **Inact** memory is only used by the pageout process to determine which pool of pages to free first at the time of a memory shortage.

The pageout process first scans the **Inact** list and checks whether the pages on this list have been accessed since the time they have been listed here. The pages that have been accessed are moved from the **Inact** list to the **Active** list. On the other hand, pages that have not been accessed become prime candidates to be freed by the pageout process. If the pageout process cannot produce enough free pages from the **Inact** list, pages from the **Active** list are freed up.

Because the pageout process runs only when the system is under memory pressure, the pages on the **Inact** list remain untouched – even if they have not been accessed recently – when the amount of **Free** memory is adequate.

How do I interpret memory numbers displayed in the show task memory command output?

The **show task memory** command provides a comprehensive picture of the memory utilization for routing protocol tasks on the Routing Engine. The routing protocol process is the main task that uses Routing Engine memory.

To check routing process memory usage, enter the **show task memory** command.

```
user@host> show task memory
Memory          Size (kB)  %Available  When
Currently In Use:    29417      3%         now
Maximum Ever Used:   33882      4%         00/02/11 22:07:03
Available:          756281    100%        now
```

[Table 5 on page 58](#) describes the output fields for the **show task memory** command. Output fields are listed in the approximate order in which they appear.

Table 5: show task memory Output Fields

Field Name	Field Description
Memory Currently In Use	Memory currently in use. Dynamically allocated memory plus the DATA segment memory in kilobytes.
Memory Maximum Ever Used	Maximum memory ever used.
Memory Available	Memory currently available.

The **show task memory** command does not display all the memory used by the routing protocol process. This value does not account for the memory used for the **TEXT** and

STACK segments, or the memory used by the routing protocol process's internal memory manager. The **show task memory** command also does not include the memory which has been deactivated by the routing protocol process, although some or all of that deactivated memory has not actually been freed by the kernel.

Why is the Memory Currently In Use value less than the RES value?

The **show task memory** command displays a **Memory Currently In Use** value measured in kilobytes. This value is the dynamically allocated memory plus the **DATA** segment memory. The **show system processes extensive** command displays a **RES** value measured in kilobytes. This value represents the amount of process memory resident in the physical memory. This is also known as RSS or Resident Set Size.

The **Memory Currently In Use** value does not account for all of the memory that the routing protocol process uses. This value does not include the memory used for the **TEXT** and the **STACK** segments, and a small percentage of memory used by the routing protocol process's internal memory manager. The **show task memory** command also does not include the memory which has been deactivated by the routing protocol process, although some or all of that deactivated memory has not actually been freed by the kernel.

Any amount of memory deactivated by the routing protocol process might still be considered part of the **RES** value. Generally, the kernel defers the actual freeing of deactivated memory until there is a memory shortage. This can lead to large discrepancies between the **Memory Currently In Use** value and the **RES** value.

Routing Protocol Process Memory Swapping FAQs

This section presents frequently asked questions and answers related to the memory swapping of the routing protocol process from the Routing Engine memory to the hard disk memory.

Why does the system start swapping when I try to perform a core dump using the request system core-dumps command?

The **request system core-dumps** command displays a list of system core files created when the device has failed. This command can be useful for diagnostic purposes. Each list item includes the file permissions, number of links, owner, group, size, modification date, path, and filename. You can use the **core-filename** option and the **core-file-info**, **brief**, and **detail** options to display more information about the specified core dump files.

You can use the **request system core-dumps** command to perform a non-fatal core dump without aborting the routing protocol process. To do this, the routing protocol process is forked, generating a second copy, and then aborted. This process can double the memory consumed by the two copies of the routing protocol process, pushing the system into swap.

Why does the show system processes extensive command show that memory is swapped to disk even though there is plenty of free memory?

Memory can remain swapped out indefinitely if it is not accessed again. Therefore, the **show system processes extensive** command shows that memory is swapped to disk even though there is plenty of free memory. Such a situation is not unusual.

Troubleshooting the Routing Protocol Process FAQs

This section presents frequently asked questions and answers related to a shortage of memory and memory leakage by the routing protocol process.

What does the RPD_OS_MEMHIGH message mean?

The **RPD_OS_MEMHIGH** message is written into the system message file if the routing protocol process is running out of memory. This message alerts you that the routing protocol process is using the indicated amount and percentage of Routing Engine memory, which is considered excessive. This message is generated either because the routing protocol process is leaking memory or the use of system resources is excessive, perhaps because routing filters are not configured properly or the configured network topology is very complex.

When the memory utilization for the routing protocol process is using all available Routing Engine DRAM memory or reaches the maximum memory limit, a message of the following form is written every minute in the syslog message file:

RPD_OS_MEMHIGH: Using 188830 KB of memory, 100 percent of available

This message includes the amount (in kilobytes), the percentage, or both of the available memory in use.

This message should not appear under normal conditions, as any further memory allocations usually require a portion of existing memory to be written to swap. As a recommended solution, increase the amount of RAM in the Routing Engine. For more information, see <http://kb.juniper.net/InfoCenter/index?page=content&id=KB14186>.

What can I do when there is a memory shortage even after a swap?

We do not recommend that the system operate in this state, notwithstanding the existence of swap. The protocols that run in the routing protocol process usually have a real-time requirement that cannot reliably withstand the latency of being swapped to hard disk. If the memory shortage has not resulted from a memory leak, then either a reduction in the memory usage or an upgrade to a higher memory-capacity Routing Engine is required.

What is the task_timer?

The source of a routing protocol process memory leak can usually be identified by dumping the timers for each task. You can use the **show task *task-name*** command to display routing protocol tasks on the Routing Engine. Tasks can be baseline tasks performed regardless of the device's configuration, and other tasks that depend on the device configuration.

For more information, see the show task command in the *Junos OS System Basics and Services Command Reference*.

Related Documentation

- [Routing Protocol Process Memory FAQ Overview on page 53](#)

PART 4

Index

- [Index on page 63](#)

Index

Symbols

#, comments in configuration statements.....	xii
(), in syntax descriptions.....	xii
< >, in syntax descriptions.....	xii
[], in configuration statements.....	xii
{ }, in configuration statements.....	xii
(pipe), in syntax descriptions.....	xii

A

active routes.....	6, 15
administrative distance.....	18
aggregate routes	
preferences.....	18
alternate preferences.....	6
always-compare-med option.....	15
as-path-ignore	
usage guidelines.....	15

B

BGP	
administrative distance.....	18
preferences.....	18
braces, in configuration statements.....	xii
brackets	
angle, in syntax descriptions.....	xii
square, in configuration statements.....	xii

C

cisco-non-deterministic option.....	15
comments, in configuration statements.....	xii
conventions	
text and syntax.....	xi
curly braces, in configuration statements.....	xii
customer support.....	xiii
contacting JTAC.....	xiii

D

documentation	
comments on.....	xiii

E

equal-cost paths.....	6, 7
external-router-id option.....	15

F

font conventions.....	xi
forwarding table	
overview.....	4
synchronizing.....	5

G

generated routes	
preferences.....	18

I

inet.0 routing table.....	4
inet.1 routing table.....	4
inet.2 routing table.....	4
inet.3 routing table.....	4
inet6.0 routing table.....	4
instance-name.inet.0 routing table.....	4
IPv6	
addressing.....	9
representation.....	10
structure.....	11
types.....	10
benefits.....	8
header fields.....	9
IS-IS	
preferences.....	18

L

load sharing.....	7
-------------------	---

M

manuals	
comments on.....	xiii
med-plus-igp statement	
usage guidelines.....	15
mpls.0 routing table.....	4
multiple active routes.....	6

N

neighbor discovery	
configuration statements.....	38

O

OSPF	
preferences.....	18

P

parentheses, in syntax descriptions.....	xii
path-selection statement	
usage guidelines.....	15
preferences	
active routes.....	6, 15
aggregate routes	
generated routes.....	18
alternate preferences.....	6
default.....	18
IS-IS.....	18
modifying	
with configuration statements.....	18
overview.....	6
RIP.....	18
static routes.....	18
tie-breaker preferences.....	6

R

redirected routes.....	18
RIP	
preferences.....	18
routing protocol databases.....	3
routing protocol process memory	
faq.....	54
routing tables	
inet.0.....	4
inet.1.....	4
inet.2.....	4
inet.3.....	4
inet6.0.....	4
instance-name.init.0.....	4
mpls.0.....	4
overview.....	4
synchronizing.....	5
rpd	
faq.....	54
rpd memory	
utilization.....	54
RSVP	
preferences.....	18

S

static routes	
preferences.....	18
support, technical See technical support	
synchronizing routing information.....	5
syntax conventions.....	xi

T

technical support	
contacting JTAC.....	xiii
tie-breaker preferences.....	6
transmit-interval statement	
BFD.....	25