JUNIPER NETWORKS

Junos® OS

Routing Policy Configuration Guide

Release
12.1

Published: 2012-03-13

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

Junos® OS Routing Policy Configuration Guide
12.1

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

List of Figures

List of Tables

About the Documentation

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:

- M Series
- T Series
- MX 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.xsl;
       }
     }
   }
   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.xsl; }
   ```

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 xix 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 xix 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
Italic text like this	• Introduces important new terms. • Identifies book names. • Identifies RFC and Internet draft titles.	• A policy *term* is a named structure that defines match conditions and actions. • *Junos OS System Basics Configuration Guide* • RFC 1997, *BGP Communities Attribute*
Italic text like this	Represents variables (options for which you substitute a value) in commands or configuration statements.	Configure the machine's domain name: [edit] root@# **set system domain-name** *domain-name*
Text like this	Represents names of configuration statements, commands, files, and directories; interface names; configuration hierarchy levels; or labels on routing platform components.	• To configure a stub area, include the **stub** statement at the **[edit protocols ospf area area-id]** hierarchy level. • The console port is labeled **CONSOLE**.
< > (angle brackets)	Enclose optional keywords or variables.	**stub** <**default-metric** *metric*>**;**
| (pipe symbol)	Indicates a choice between the mutually exclusive keywords or variables on either side of the symbol. The set of choices is often enclosed in parentheses for clarity.	**broadcast | multicast** (***string1*** **|** ***string2*** **|** ***string3***)
# (pound sign)	Indicates a comment specified on the same line as the configuration statement to which it applies.	**rsvp { # Required for dynamic MPLS only**
[ ] (square brackets)	Enclose a variable for which you can substitute one or more values.	**community name members [** ***community-ids*** **]**

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

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

Documentation Feedback

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

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

Requesting Technical Support

Technical product support is available through the Juniper Networks Technical Assistance Center (JTAC). If you are a customer with an active J-Care or JNASC support contract, or are covered under warranty, and need post-sales technical support, you can access our tools and resources online or open a case with JTAC.

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

Self-Help Online Tools and Resources

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

- Find CSC offerings: http://www.juniper.net/customers/support/
- Search for known bugs: http://www2.juniper.net/kb/
- Find product documentation: http://www.juniper.net/techpubs/
- Find solutions and answer questions using our Knowledge Base: http://kb.juniper.net/
- Download the latest versions of software and review release notes: http://www.juniper.net/customers/csc/software/
- Search technical bulletins for relevant hardware and software notifications: https://www.juniper.net/alerts/

- Join and participate in the Juniper Networks Community Forum: http://www.juniper.net/company/communities/
- Open a case online in the CSC Case Management tool: http://www.juniper.net/cm/

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

Opening a Case with JTAC

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

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

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

PART 1

Overview

CHAPTER 1

Introduction to Policy Framework

Policy Framework Overview

The Junos OS provides a *policy framework*, which is a collection of Junos OS policies that allows you to control flows of routing information and packets. The policy framework is composed of the following policies:

- Routing policy—Allows you to control the routing information between the routing protocols and the routing tables and between the routing tables and the forwarding table
- Firewall filter policy—Allows you to control packets transiting the router to a network destination and packets destined for and sent by the router

NOTE: The term *firewall filter policy* is used here to emphasize that a firewall filter is a policy and shares some fundamental similarities with a routing policy. However, when referring to a firewall filter policy in the rest of this manual, the term *firewall filter* is used.

Router Flows Affected by Policies

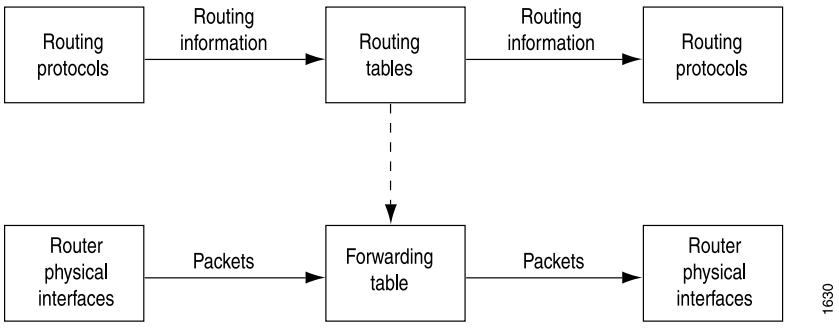
The Junos OS policies affect the following router flows:

- Flow of routing information between the routing protocols and the routing tables and between the routing tables and the forwarding table. The Routing Engine handles this flow. *Routing information* is the information about routes learned by the routing protocols from a router's neighbors. This information is stored in routing tables and is subsequently advertised by the routing protocols to the router's neighbors. Routing policies allow you to control the flow of this information.
- Flow of data packets in and out of the router's physical interfaces. The Packet Forwarding Engine handles this flow. *Data packets* are chunks of data that transit the router as they are being forwarded from a source to a destination. When a router receives a data packet on an interface, it determines where to forward the packet by looking in the forwarding table for the best route to a destination. The router then forwards the data packet toward its destination through the appropriate interface. Firewall filters allow you to control the flow of these data packets.
- Flow of local packets from the router's physical interfaces and to the Routing Engine. The Routing Engine handles this flow. *Local packets* are chunks of data that are destined for or sent by the router. Local packets usually contain routing protocol data, data for IP services such as Telnet or SSH, and data for administrative protocols such as the Internet Control Message Protocol (ICMP). When the Routing Engine receives a local packet, it forwards the packet to the appropriate process or to the kernel, which are both part of the Routing Engine, or to the Packet Forwarding Engine. Firewall filters allow you to control the flow of these local packets.

NOTE: **In the rest of this chapter, the term *packets* refers to both data and local packets unless explicitly stated otherwise.**

Figure 1 on page 4 illustrates the flows through the router. Although the flows are very different from each other, they are also interdependent. Routing policies determine which routes are placed in the forwarding table. The forwarding table, in turn, has an integral role in determining the appropriate physical interface through which to forward a packet.

Figure 1: Flows of Routing Information and Packets

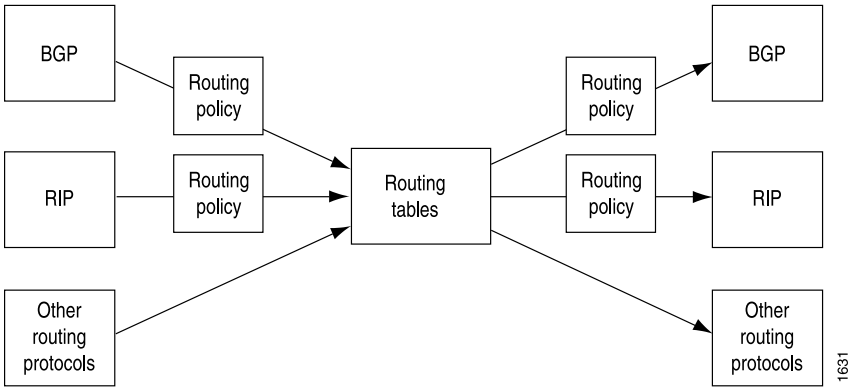


You can configure routing policies to control which routes the routing protocols place in the routing tables and to control which routes the routing protocols advertise from the routing tables (see Figure 2 on page 5). The routing protocols advertise active routes only from the routing tables. (An *active route* is a route that is chosen from all routes in the routing table to reach a destination.)

You can also use routing policies to do the following:

- Change specific route characteristics, which allow you to control which route is selected as the active route to reach a destination. In general, the active route is also advertised to a router's neighbors.
- Change to the default BGP route flap-damping values.
- Perform per-packet load balancing.
- Enable class of service (CoS).

Figure 2: Routing Policies to Control Routing Information Flow

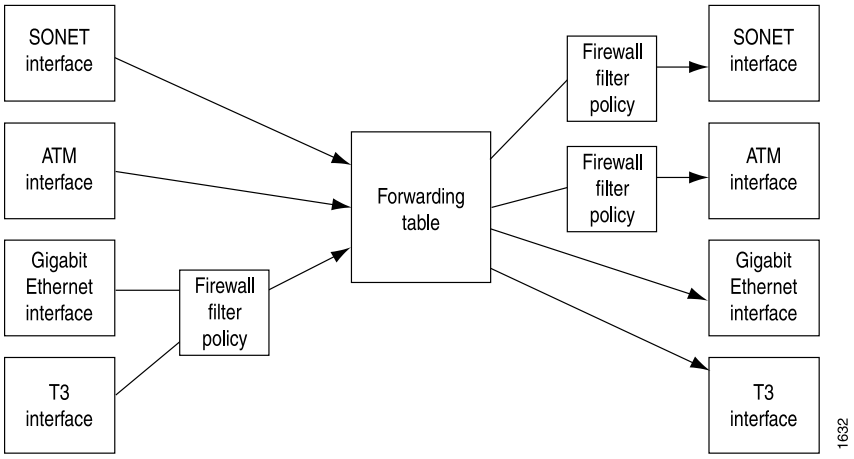


You can configure firewall filters to control the following aspects of packet flow (see Figure 3 on page 6):

- Which data packets are accepted on and transmitted from the physical interfaces. To control the flow of data packets, you apply firewall filters to the physical interfaces.
- Which local packets are transmitted from the physical interfaces and to the Routing Engine. To control local packets, you apply firewall filters on the loopback interface, which is the interface to the Routing Engine.

Firewall filters provide a means of protecting your router from excessive traffic transiting the router to a network destination or destined for the Routing Engine. Firewall filters that control local packets can also protect your router from external incidents such as denial-of-service attacks.

Figure 3: Firewall Filters to Control Packet Flow



Policy Architecture

A *policy* is a mechanism in the Junos OS policy framework that allows you to configure criteria against which something can be compared and an action that is performed if the criteria are met.

All policies in the Junos OS policy framework share the following architecture and configuration fundamentals:

- Control Points on page 6
- Policy Components on page 7
- Default Policies and Actions on page 8
- Configuration Tasks on page 8
- Policy Configuration Recommendations on page 8

NOTE: This section highlights the fundamental architecture that all policies share. Note, however, that the implementation details of routing policies and firewall filters are very different. For information about these differences, see "Comparison of Routing Policies and Firewall Filters" on page 9.

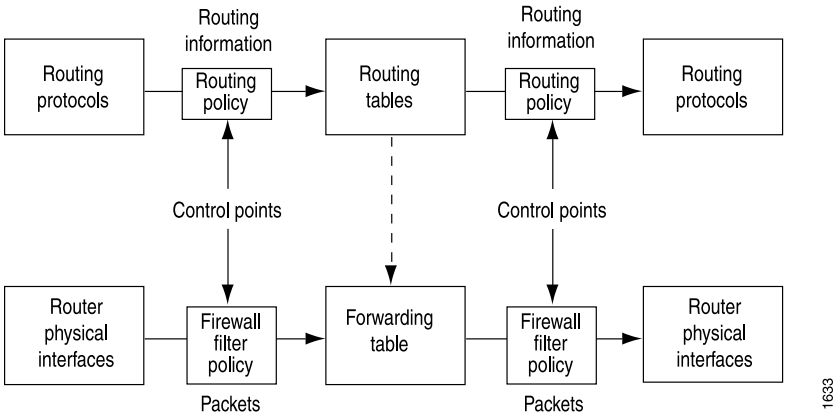
Control Points

All policies provide two points at which you can control routing information or packets through the router (see Figure 4 on page 7). These control points allow you to control the following:

- Routing information before and after it is placed in the routing table.
- Data packets before and after a forwarding table lookup.

- Local packets before and after they are received by the Routing Engine. (Figure 4 on page 7 appears to depict only one control point but because of the bidirectional flow of the local packets, two control points actually exist.)

Figure 4: Policy Control Points



Because there are two control points, you can configure policies that control the routing information or data packets before and after their interaction with their respective tables, and policies that control local packets before and after their interaction with the Routing Engine. *Import routing policies* control the routing information that is placed in the routing tables, whereas *export routing policies* control the routing information that is advertised from the routing tables. *Input firewall filters* control packets that are received on a router interface, whereas *output firewall filters* control packets that are transmitted from a router interface.

Policy Components

All policies are composed of the following components that you configure:

- *Match conditions*—Criteria against which a route or packets are compared. You can configure one or more criteria. If all criteria match, one or more actions are applied.
- *Actions*—What happens if all criteria match. You can configure one or more actions.
- *Terms*—Named structures in which match conditions and actions are defined. You can define one or more terms.

For more information about these concepts and how they fit into the context of their respective policies, see "Configuring a Routing Policy" on page 18 and Stateless Firewall Filter Components.

The policy framework software evaluates each incoming and outgoing route or packet against the match conditions in a term. If the criteria in the match conditions are met, the defined action is taken.

In general, the policy framework software compares the route or packet against the match conditions in the first term in the policy, then goes on to the next term, and so on. (For specific information about when the evaluation process ends for each policy, see

"Comparison of Routing Policies and Firewall Filters" on page 9.) Therefore, the order in which you arrange terms in a policy is relevant.

The order of match conditions within a term is not relevant because a route or packet must match all match conditions in a term for an action to be taken.

Default Policies and Actions

If an incoming or outgoing route or packet arrives and there is no explicitly configured policy related to the route or to the interface upon which the packet arrives, the action specified by the default policy is taken. A *default policy* is a rule or a set of rules that determine whether the route is placed in or advertised from the routing table, or whether the packet is accepted into or transmitted from the router interface.

All policies also have default actions in case one of the following situations arises during policy evaluation:

- A policy does not specify a match condition.
- A match occurs, but a policy does not specify an action.
- A match does not occur with a term in a policy and subsequent terms in the same policy exist.
- A match does not occur by the end of a policy.

Configuration Tasks

All policies share a two-step configuration process:

- Define the policy—Define the policy components. The components include criteria against which routes or packets are compared and actions that are performed if the criteria are met. For more information, see "Policy Components" on page 7.
- Apply the policy—Apply the policy to whatever moves the routing information or packets through the router, for example, the routing protocol or the router interface.

NOTE: **A defined policy does not take effect until you apply it.**

Policy Configuration Recommendations

The Junos OS policy architecture is simple and straightforward. However, the actual implementation of each policy adds layers of complexity to the policy as well as adding power and flexibility to your router's capabilities. Configuring a policy has a major impact on the flow of routing information or packets within and through the router. For example, you can configure a routing policy that does not allow routes associated with a particular customer to be placed in the routing table. As a result of this routing policy, the customer routes are not used to forward data packets to various destinations and the routes are not advertised by the routing protocol to neighbors.

Before configuring a policy, determine what you want to accomplish with it and thoroughly understand how to achieve your goal using the various match conditions and actions. Also, make certain that you understand the default policies and actions for the policy you are configuring.

Comparison of Routing Policies and Firewall Filters

Although routing policies and firewall filters share an architecture, as described in "Policy Architecture" on page 6, their purposes, implementation, and configuration are different. Table 3 on page 9 describes their purposes. Table 4 on page 9 compares the implementation details for routing policies and firewall filters, highlighting the similarities and differences in their configuration.

Table 3: Purpose of Routing Policies and Firewall Filters

Policies	Source	Policy Purpose
Routing policies	Routing information is generated by internal networking peers.	To control the size and content of the routing tables, which routes are advertised, and which routes are considered the best to reach various destinations.
Firewall filters	Packets are generated by internal and external devices through which hostile attacks can be perpetrated.	To protect your router and network from excessive incoming traffic or hostile attacks that can disrupt network service, and to control which packets are forwarded from which router interfaces.

Table 4: Implementation Differences Between Routing Policies and Firewall Filters

Policy Architecture	Routing Policy Implementation	Firewall Filter Implementation
Control points	Control routing information that is placed in the routing table with an import routing policy and advertised from the routing table with an export routing policy.	Control packets that are accepted on a router interface with an input firewall filter and that are forwarded from an interface with an output firewall filter.
Configuration tasks: • Define policy • Apply policy	Define a policy that contains terms, match conditions, and actions. Apply one or more export or import policies to a routing protocol. You can also apply a *policy expression*, which uses Boolean logical operators with multiple import or export policies. You can also apply one or more export policies to the forwarding table.	Define a policy that contains terms, match conditions, and actions. Apply one input or output firewall filter to a physical interface or physical interface group to filter data packets received by or forwarded to a physical interface (on routing platforms with an Internet Processor II application-specific integrated circuit [ASIC] only). You can also apply one input or output firewall filter to the routing platform's loopback interface, which is the interface to the Routing Engine (on all routing platforms). This allows you to filter local packets received by or forwarded from the Routing Engine.

Table 4: Implementation Differences Between Routing Policies and Firewall Filters *(continued)*

Policy Architecture	Routing Policy Implementation	Firewall Filter Implementation
Terms	Configure as many terms as desired. Define a name for each term. Terms are evaluated in the order in which you specify them. Evaluation of a policy ends after a packet matches the criteria in a term and the defined or default policy action of accept or reject is taken. The route is not evaluated against subsequent terms in the same policy or subsequent policies.	Configure as many terms as desired. Define a name for each term. Terms are evaluated in the order in which you specify them. Evaluation of a firewall filter ends after a packet matches the criteria in a term and the defined or default action is taken. The packet is not evaluated against subsequent terms in the firewall filter.
Match conditions	Specify zero or more criteria that a route must match. You can specify criteria based on source, destination, or properties of a route. You can also specify the following match conditions, which require more configuration: • Autonomous system (AS) path expression—A combination of AS numbers and regular expression operators. • Community—A group of destinations that share a common property. • Prefix list—A named list of prefixes. • Route list—A list of destination prefixes. • Subroutine—A routing policy that is called repeatedly from other routing policies.	Specify zero or more criteria that a packet must match. You must match various fields in the packet's header. The fields are grouped into the following categories: • Numeric values, such as port and protocol numbers. • Prefix values, such as IP source and destination prefixes. • Bit-field values—Whether particular bits in the fields are set, such as IP options, Transmission Control Protocol (TCP) flags, and IP fragmentation fields. You can specify the fields using Boolean logical operators.

Table 4: Implementation Differences Between Routing Policies and Firewall Filters *(continued)*

Policy Architecture	Routing Policy Implementation	Firewall Filter Implementation
Actions	Specify zero or one action to take if a route matches all criteria. You can specify the following actions: • Accept—Accept the route into the routing table, and propagate it. After this action is taken, the evaluation of subsequent terms and policies ends. • Reject—Do not accept the route into the routing table, and do not propagate it. After this action is taken, the evaluation of subsequent terms and policies ends. In addition to the preceding actions, you can also specify zero or more of the following types of actions: • Next term—Evaluate the next term in the routing policy. • Next policy—Evaluate the next routing policy. • Actions that manipulate characteristics associated with a route as the routing protocol places it in the routing table or advertises it from the routing table. • Trace action, which logs route matches.	Specify zero or one action to take if a packet matches all criteria. (We recommend that you always explicitly configure an action.) You can specify the following actions: • Accept—Accept a packet. • Discard—Discard a packet silently, without sending an ICMP message. • Reject—Discard a packet, and send an ICMP destination unreachable message. • Routing instance—Specify a routing table to which packets are forwarded. • Next term—Evaluate the next term in the firewall filter. In addition to zero or the preceding actions, you can also specify zero or more action modifiers. You can specify the following action modifiers: • Count—Add packet to a count total. • Forwarding class—Set the packet forwarding class to a specified value from 0 through 3. • IPsec security association—Used with the source and destination address match conditions, specify an IP Security (IPsec) security association (SA) for the packet. • Log—Store the header information of a packet on the Routing Engine. • Loss priority—Set the packet loss priority (PLP) bit to a specified value, 0 or 1. • Policer—Apply rate-limiting procedures to the traffic. • Sample—Sample the packet traffic. • Syslog—Log an alert for the packet.

Table 4: Implementation Differences Between Routing Policies and Firewall Filters *(continued)*

Policy Architecture	Routing Policy Implementation	Firewall Filter Implementation

Default policies and actions	If an incoming or outgoing route arrives and a policy related to the route is not explicitly configured, the action specified by the default policy for the associated routing protocol is taken.
	
	The following default actions exist for routing policies:
	
	• If a policy does not specify a match condition, all routes evaluated against the policy match.
	• If a match occurs but the policy does not specify an accept, reject, next term, or next policy action, one of the following occurs:
	  ◦ The next term, if present, is evaluated.
	  ◦ If no other terms are present, the next policy is evaluated.
	  ◦ If no other policies are present, the action specified by the default policy is taken.
	• If a match does not occur with a term in a policy and subsequent terms in the same policy exist, the next term is evaluated.
	• If a match does not occur with any terms in a policy and subsequent policies exist, the next policy is evaluated.
	• If a match does not occur by the end of a policy and no other policies exist, the accept or reject action specified by the default policy is taken.
		If an incoming or outgoing packet arrives on an interface and a firewall filter is not configured for the interface, the default policy is taken (the packet is accepted).
		
		The following default actions exist for firewall filters:
		
		• If a firewall filter does not specify a match condition, all packets are considered to match.
		• If a match occurs but the firewall filter does not specify an action, the packet is accepted.
		• If a match occurs, the defined or default action is taken and the evaluation ends. Subsequent terms in the firewall filter are not evaluated, unless the next term action is specified.
		• If a match does not occur with a term in a firewall filter and subsequent terms in the same filter exist, the next term is evaluated.
		• If a match does not occur by the end of a firewall filter, the packet is discarded.

CHAPTER 2

Introduction to Routing Policy

Routing Policy Overview

All routing protocols store their routing information in routing tables. From these tables, the routing protocols calculate the best route to each destination and place these routes in a forwarding table. These routes are then used to forward routing protocol traffic toward a destination, and they can be advertised to neighbors using one or more routing protocols.

NOTE: Instead of referring to the multiple routing tables that the Junos OS maintains, the discussion in the rest of this chapter assumes the inet.0 routing table unless explicitly stated otherwise. By default, the Junos OS stores unicast IP version 4 (IPv4) routes in the inet.0 routing table. For information about all the routing tables, see "Routing Tables Affected by Routing Policies" on page 228.

In general, the routing protocols place all their routes in the routing table and advertise a limited set of routes from the routing table. The general rules for handling the routing information between the routing protocols and the routing table are known as the *routing policy framework*.

The routing policy framework is composed of default rules for each routing protocol that determine which routes the protocol places in the routing table and advertises from the routing table. The default rules for each routing protocol are known as *default routing policies*.

You can create routing policies to preempt the default policies, which are always present. A *routing policy* is a mechanism in the Junos OS that allows you to modify the routing policy framework to suit your needs. You can create and implement your own routing policies to do the following:

- Control which routes a routing protocol places in the routing table.
- Control which active routes a routing protocol advertises from the routing table. An *active route* is a route that is chosen from all routes in the routing table to reach a destination.
- Manipulate the route characteristics as a routing protocol places it in the routing table or advertises it from the routing table.

You can manipulate the route characteristics to control which route is selected as the active route to reach a destination. The active route is placed in the forwarding table and used to forward traffic toward the route's destination. In general, the active route is also advertised to a router's neighbors.

To create a routing policy, you must define the policy and apply it. You define the policy by specifying the criteria that a route must match and the actions to perform if a match occurs. You then apply the policy to a routing protocol or to the forwarding table.

NOTE: Before you create your routing policies, we recommend that you read through this entire section to become familiar with the terminology, concepts, and configuration guidelines.

In Junos OS Release 9.5 and later, you can configure routing policies and certain routing policy objects in a dynamic database that is not subject to the same verification required by the standard configuration database. As a result, you can quickly commit these routing policies and policy objects, which can be referenced and applied in the standard

configuration as needed. BGP is the only protocol to which you can apply routing policies that reference policies configured in the dynamic database. After a routing policy based on the dynamic database is configured and committed in the standard configuration, you can quickly make changes to existing routing policies by modifying policy objects in the dynamic database. Because the Junos OS does not validate configuration changes to the dynamic database, when you use this feature, you should test and verify all configuration changes before committing them. For more information about configuring dynamic routing policies, see "Configuring Dynamic Routing Policies" on page 70.

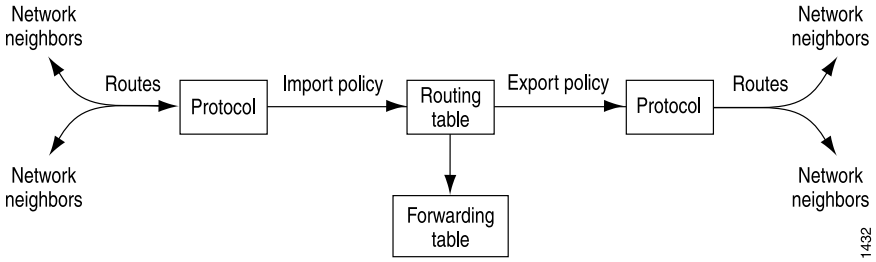
Importing and Exporting Routes

Two terms—*import* and *export*—explain how routes move between the routing protocols and the routing table (see Figure 5 on page 15):

- When the Routing Engine places the routes of a routing protocol into the routing table, it is *importing* routes into the routing table.
- When the Routing Engine uses active routes from the routing table to send a protocol advertisement, it is *exporting* routes from the routing table.

NOTE: The process of moving routes between a routing protocol and the routing table is described always *from the point of view of the routing table*. That is, routes are *imported into* a routing table from a routing protocol and they are *exported from* a routing table to a routing protocol. Remember this distinction when working with routing policies.

Figure 5: Importing and Exporting Routes



When evaluating routes for export, the Routing Engine uses only active routes from the routing table. For example, if a routing table contains multiple routes to the same destination and one route has a preferable metric, only that route is evaluated. In other words, an export policy does not evaluate all routes; it evaluates only those routes that a routing protocol is allowed to advertise to a neighbor.

NOTE: By default, BGP advertises active routes. However, you can configure BGP to advertise *inactive routes*, which go to the same destination as other routes but have less preferable metrics.

"Protocols That Can Be Imported to and Exported from the Routing Table" on page 227 lists the routing protocols from which the routing table can import routes and to which the routing table can export routes. That topic also lists direct and explicitly configured routes, which for the purposes of this table are considered a pseudoprotocol. (An *explicitly configured route* is a route that you have configured. *Direct routes* are not explicitly configured; they are created as a result of IP addresses being configured on an interface.) Explicitly configured routes include aggregate, generated, local, and static routes. (An *aggregate route* is a route that distills groups of routes with common addresses into one route. A *generated route* is a route used when the routing table has no information about how to reach a particular destination. A *local route* is an IP address assigned to a router interface. A *static route* is an unchanging route to a destination.)

The policy framework software treats direct and explicitly configured routes as if they are learned through routing protocols; therefore, they can be imported into the routing table. Routes cannot be exported from the routing table to the pseudoprotocol, because this protocol is not a real routing protocol. However, aggregate, direct, generated, and static routes can be exported from the routing table to routing protocols, whereas local routes cannot.

For information about the default routing policies for each routing protocol, see "Default Import and Export Policies for Protocols" on page 229. For information about the import and export routing policies supported for each routing protocol and the level at which you can apply these policies, see "Protocols That Can Be Imported to and Exported from the Routing Table" on page 227.

Default Routing Policies and Actions

You must be familiar with the default routing policies to know when you need to modify them to suit your needs. "Default Import and Export Policies for Protocols" on page 229 summarizes the default routing policies for each routing protocol that imports and exports routes. The actions in the default routing policies are taken if you have not explicitly configured a routing policy. This table also shows direct and explicitly configured routes, which for the purposes of this table are considered a pseudoprotocol. Explicitly configured routes include aggregate, generated, and static routes.

When multiple routes for a destination exist in the routing table, the protocol selects an active route and that route is placed in the appropriate routing table. For equal-cost routes, the Junos OS places multiple next hops in the appropriate routing table.

When a protocol is exporting routes from the routing table, it exports active routes only. This applies to actions specified by both default and user-defined export policies.

You cannot change the default import policy for the link-state protocols IS-IS and OSPF. As link-state protocols, IS-IS and OSPF exchange routes between systems within an autonomous system (AS). All routers and systems within an AS must share the same link-state database, which includes routes to reachable prefixes and the metrics associated with the prefixes. If an import policy is configured and applied to IS-IS or OSPF, some routes might not be learned or advertised or the metrics for learned routes might be altered, which would make a consistent link-state database impossible.

The default export policy for IS-IS and OSPF protocols is to reject everything. These protocols do not actually export their internally learned routes (the directly connected routes on interfaces that are running the protocol). Both IS-IS and OSPF protocols use a procedure called flooding to announce local routes and any routes learned by the protocol. The flooding procedure is internal to the protocol, and is unaffected by the policy framework. Exporting can be used only to announce information from other protocols, and the default is not to do so.

For information about the routing protocols from which the routing table can import routes and to which routing protocols the routing table can export routes, see "Protocols That Can Be Imported to and Exported from the Routing Table" on page 227. For information about the user-defined import and export policies supported for each routing protocol and the level at which you can apply these policies, see "Protocol Support for Import and Export Policies" on page 231.

The following default actions are taken if the following situations arise during policy evaluation:

- If a policy does not specify a match condition, all routes evaluated against the policy match.
- If a match occurs but the policy does not specify an accept, reject, next term, or next policy action, one of the following occurs:
  - The next term, if present, is evaluated.
  - If no other terms are present, the next policy is evaluated.
  - If no other policies are present, the action specified by the default policy is taken.
- If a match does not occur with a term in a policy and subsequent terms in the same policy exist, the next term is evaluated.
- If a match does not occur with any terms in a policy and subsequent policies exist, the next policy is evaluated.
- If a match does not occur by the end of a policy or all policies, the accept or reject action specified by the default policy is taken.

The default import policy is always the same: accept all routes learned from the protocol. "Default Import and Export Policies for Protocols" on page 229 includes information about the routing tables used by each protocol.

Creating Routing Policies

The following are typical circumstances under which you might want to preempt the default routing policies in the routing policy framework by creating your own routing policies:

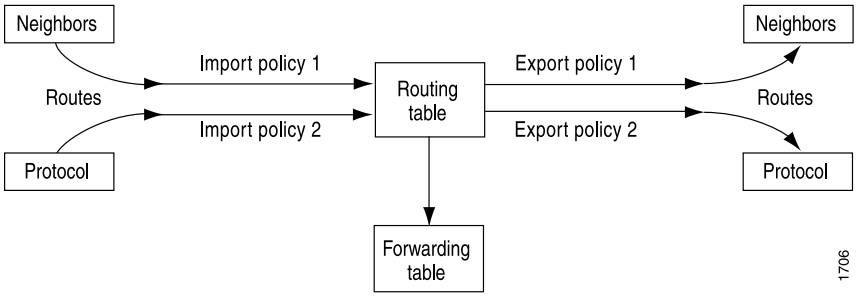
- You do not want a protocol to import all routes into the routing table. If the routing table does not learn about certain routes, they can never be used to forward packets and they can never be redistributed into other routing protocols.
- You do not want a routing protocol to export all the active routes it learns.

- You want a routing protocol to announce active routes learned from another routing protocol, which is sometimes called *route redistribution*.
- You want to manipulate route characteristics, such as the preference value, AS path, or community. You can manipulate the route characteristics to control which route is selected as the active route to reach a destination. In general, the active route is also advertised to a router's neighbors.
- You want to change the default BGP route flap-damping parameters.
- You want to perform per-packet load balancing.
- You want to enable class of service (CoS).

Configuring a Routing Policy

As shown in Figure 6 on page 18, you use *import routing policies* to control which routes routing protocols place in the routing table, and *export routing policies* to control which routes a routing protocol advertises from the routing table to its neighbors.

Figure 6: Importing and Exporting Routing Policies

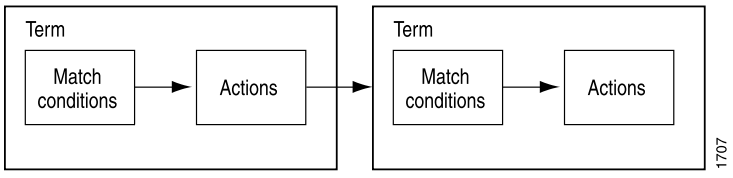


To create a routing policy, you must define the following components:

- *Match conditions*—Criteria that a route must match. If a route matches all of the criteria, one or more actions are applied to the route.
- *Actions*—What to do if a route matches. The actions can specify whether to accept or reject the route, control how a series of policies is evaluated, and manipulate the characteristics associated with a route. You can configure one or more actions.

You typically define match conditions and actions within a *term*. Figure 7 on page 18 shows the routing policy components, including the term.

Figure 7: Routing Policy Components



After defining a routing policy, you then apply it to a routing protocol or to the forwarding table.

This section provides more information about creating routing policies:

Routing Policy Named Match Conditions

Some match conditions are defined separately from the routing policy and are given names. You then reference the name of the match condition in the definition of the routing policy itself. Named match conditions allow you to do the following:

- Reuse match conditions in other routing policies.
- Read configurations that include complex match conditions more easily.

Named match conditions include communities, prefix lists, and AS path regular expressions. For more information about these match conditions, see "Routing Policy Match Conditions" on page 230.

Routing Policy Actions

An *action* is what the policy framework software does if a route matches all criteria defined in a match condition. You can configure one or more actions in a term. The policy framework software supports the following types of actions:

- Flow control actions, which affect whether to accept or reject the route or whether to evaluate the next term or routing policy
- Actions that manipulate route characteristics
- Trace action, which logs route matches

Manipulating the route characteristics allows you to control which route is selected as the active route to reach a destination. In general, the active route is also advertised to a routing platform's neighbors. You can manipulate the following route characteristics: AS path, class, color, community, damping parameters, destination class, external type, next hop, load balance, local preference, metric, origin, preference, and tag.

For the numeric information (color, local preference, metric, preference, and tag), you can set a specific value or change the value by adding or subtracting a specified amount. The addition and subtraction operations do not allow the value to exceed a maximum value and drop below a minimum value.

For more information about the properties you can change and the addition and subtraction operations, see "Configuring Actions in Routing Policy Terms" on page 54.

Routing Policy Terms

A *term* is a named structure in which match conditions and actions are defined. You can define one or more terms.

In general, the policy framework software compares a route against the match conditions in the first term in the first routing policy, then goes on to the next term and the next policy if present, and so on, until an explicitly configured or default action of accept or reject is taken. Therefore, the order in which you arrange terms in a policy is relevant.

The order of match conditions in a term is not relevant, because a route must match all match conditions in a term for an action to be taken.

Applying a Routing Policy

After defining a routing policy, as discussed in "Routing Policy Match Conditions" on page 230 and "Routing Policy Actions" on page 19, you can apply it to one of the following:

- Routing protocols—BGP, DVMRP, IS-IS, LDP, MPLS, OSPF, PIM dense mode, PIM sparse mode, PIM sparse-dense mode, RIP, and RIPng
- Pseudoprotocol—Explicitly created routes, which include aggregate and generated routes
- Forwarding table

The following sections discuss the following topics:

Routing Protocol Support for Import and Export Policies

When applying routing policies to routing protocols, you must know whether each protocol supports import and export policies and the level at which you can apply these policies. "Protocol Support for Import and Export Policies" on page 231 summarizes the import and export policy support for each routing protocol. "Protocols That Can Be Imported to and Exported from the Routing Table" on page 227 also lists explicitly configured routes, which for the purposes of this table are considered a pseudoprotocol. Explicitly configured routes include aggregate and generated routes.

You can apply an import policy to aggregate and generated routes, but you cannot apply an export policy to these routes. These routes cannot be exported from the routing table to the pseudoprotocol, because this protocol is not a real routing protocol. However, aggregate and generated routes can be exported from the routing table to routing protocols.

You cannot apply import policies to the link-state protocols IS-IS and OSPF. As link-state protocols, IS-IS and OSPF exchange routes between systems within an AS. All routers and systems within an AS must share the same link-state database, which includes routes to reachable prefixes and the metrics associated with the prefixes. If an import policy is configured and applied to IS-IS or OSPF, some routes might not be learned or advertised, or the metrics for learned routes might be altered, which would make a consistent link-state database impossible.

For BGP only, you can also apply import and export policies at group and peer levels as well as at the global level. A peer import or export policy overrides a group import or export policy. A group import or export policy overrides a global import or export policy.

For example, if you define an import policy for an individual peer at the peer level and also define an import policy for the group to which it belongs, the import policy defined for the peer level only is invoked. The group import policy is not used for that peer, but it is applied to other peers in that group.

For RIP and RIPng only, you can apply import policies at the global and neighbor levels and export policies at a group level.

For information about the routing protocols from which the routing table can import routes and to which routing protocols the routing table can export routes, see "Protocols That Can Be Imported to and Exported from the Routing Table" on page 227. For information about the default routing policies for each routing protocol, see "Default Import and Export Policies for Protocols" on page 229.

Applying a Routing Policy to Routing Protocols

You can apply the following routing policy elements to a routing protocol:

- Routing policy—You can apply a single routing policy to a routing protocol.
- Chain of routing policies—You can apply multiple routing policies (chains) to a routing protocol.
- Policy expression—You can apply a policy expression to a routing protocol. A *policy expression* uses Boolean logical operators with a routing policy and routing policy chains. The logical operators establish rules by which the policy or chains are evaluated.

Applying Export Policies to the Forwarding Table

You can apply export policies to routes being exported from the routing table into the forwarding table for the following features:

- Per-packet load balancing
- CoS

For more information about per-packet load balancing, see "Configuring Per-Packet Load Balancing" on page 160.

Related Documentation

- *Junos OS Class of Service Configuration Guide*

Evaluating a Routing Policy

This section provides information about how routing policies are evaluated. It discusses the following topics:

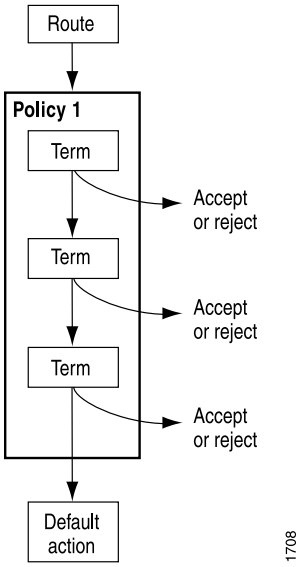
For specific information about how the various match conditions are evaluated, see "Configuring Match Conditions in Routing Policy Terms" on page 47.

How a Routing Policy Is Evaluated

Figure 8 on page 23 shows how a single routing policy is evaluated. This routing policy consists of multiple terms. Each term consists of match conditions and actions to apply to matching routes. Each route is evaluated against the policy as follows:

1. The route is evaluated against the first term. If it matches, the specified action is taken. If the action is to accept or reject the route, that action is taken and the evaluation of the route ends. If the next term action is specified, if no action is specified, or if the route does not match, the evaluation continues as described in Step 2. If the next policy action is specified, any accept or reject action specified in this term is skipped, all remaining terms in this policy are skipped, all other actions are taken, and the evaluation continues as described in Step 3.
2. The route is evaluated against the second term. If it matches, the specified action is taken. If the action is to accept or reject the route, that action is taken and the evaluation of the route ends. If the next term action is specified, if no action is specified, or if the route does not match, the evaluation continues in a similar manner against the last term. If the next policy action is specified, any accept or reject action specified in this term is skipped, all remaining terms in this policy are skipped, all other actions are taken, and the evaluation continues as described in Step 3.
3. If the route matches no terms in the routing policy or the next policy action is specified, the accept or reject action specified by the default policy is taken. For more information about the default routing policies, see "Default Routing Policies and Actions" on page 16.

Figure 8: Routing Policy Evaluation

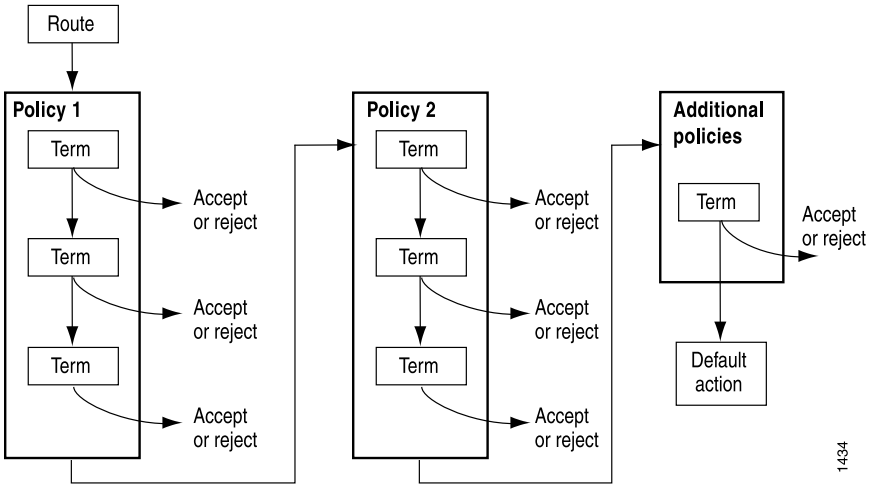


How a Routing Policy Chain Is Evaluated

Figure 9 on page 24 shows how a chain of routing policies is evaluated. These routing policies consist of multiple terms. Each term consists of match conditions and actions to apply to matching routes. Each route is evaluated against the policies as follows:

1. The route is evaluated against the first term in the first routing policy. If it matches, the specified action is taken. If the action is to accept or reject the route, that action is taken and the evaluation of the route ends. If the next term action is specified, if no action is specified, or if the route does not match, the evaluation continues as described in Step 2. If the next policy action is specified, any accept or reject action specified in this term is skipped, all remaining terms in this policy are skipped, all other actions are taken, and the evaluation continues as described in Step 3.
2. The route is evaluated against the second term in the first routing policy. If it matches, the specified action is taken. If the action is to accept or reject the route, that action is taken and the evaluation of the route ends. If the next term action is specified, if no action is specified, or if the route does not match, the evaluation continues in a similar manner against the last term in the first routing policy. If the next policy action is specified, any accept or reject action specified in this term is skipped, all remaining terms in this policy are skipped, all other actions are taken, and the evaluation continues as described in Step 3.
3. If the route does not match a term or matches a term with a next policy action in the first routing policy, it is evaluated against the first term in the second routing policy.
4. The evaluation continues until the route matches a term with an accept or reject action defined or until there are no more routing policies to evaluate. If there are no more routing policies, then the accept or reject action specified by the default policy is taken. For more information about default routing policies, see "Default Routing Policies and Actions" on page 16.

Figure 9: Routing Policy Chain Evaluation



How a Routing Policy Expression Is Evaluated

To understand how a policy expression is evaluated, you must first understand the Boolean logical operators and the associated logic used in evaluating a policy expression. For more information about policy expressions, including how they are evaluated, see "Applying Policy Expressions to Routes Exported from Routing Tables" on page 65.

How a Routing Policy Subroutine Is Evaluated

Figure 10 on page 26 shows how a subroutine is evaluated. The subroutine is included in the first term of the first routing policy in a chain. Each route is evaluated against the subroutine as follows:

1. The route is evaluated against the first term in the first routing policy. If the route does not match all match conditions specified before the subroutine, the subroutine is skipped and the next term in the routing policy is evaluated (see Step 2). If the route matches all match conditions specified before the subroutine, the route is evaluated against the subroutine. If the route matches the match conditions in any of the subroutine terms, two levels of evaluation occur in the following order:

   a. The actions in the subroutine term are evaluated. If one of the actions is accept, evaluation of the subroutine ends and a Boolean value of TRUE is returned to the calling policy. If one of the actions is reject, evaluation of the subroutine ends and FALSE is returned to the calling policy.

      If the subroutine does not specify the **accept**, **reject** or **next-policy** action, it uses the **accept** or **reject** action specified by the default policy, and the values of TRUE or FALSE are returned to the calling policy as described in the previous paragraph. (For information about what happens if a termination action is not specified in the term, see "Configuring Subroutines in Routing Policy Match Conditions" on page 146. For more information about the default routing policies, see "Default Routing Policies and Actions" on page 16.)

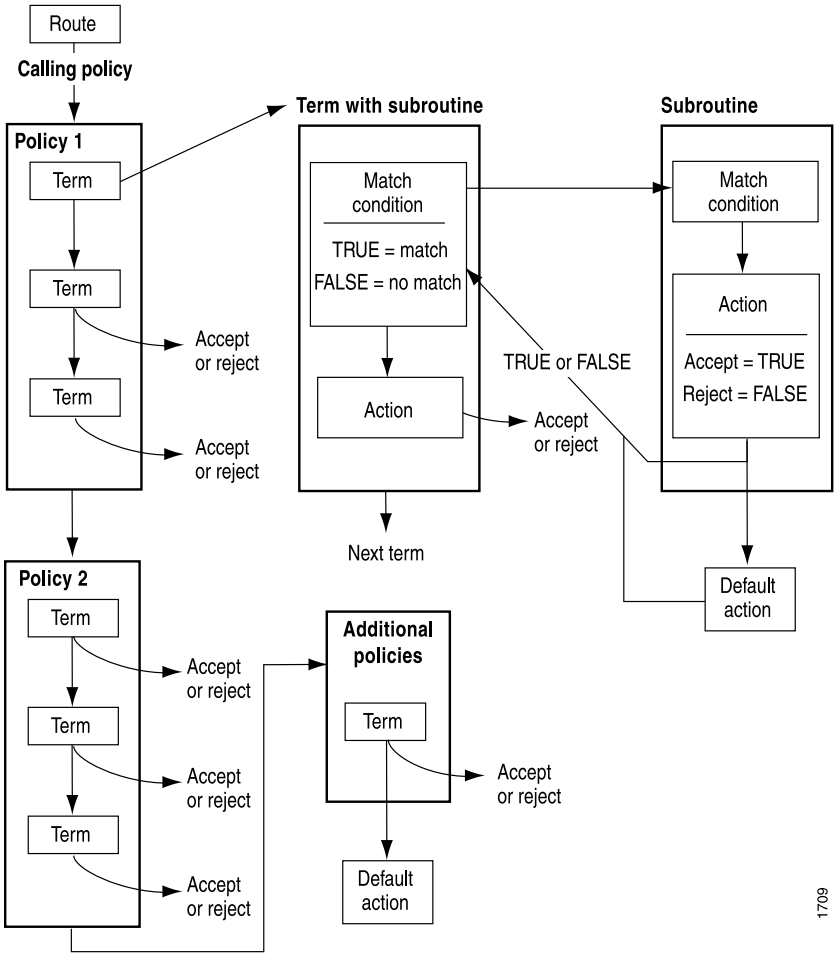
b. The calling policy's subroutine match condition is evaluated. During this part of the evaluation, TRUE equals a match and FALSE equals no match. If the subroutine returns TRUE to the calling policy, then the evaluation of the calling policy continues. If the subroutine returns FALSE to the calling policy, then the evaluation of the current term ends and the next term is evaluated.

2. The route is evaluated against the second term in the first routing policy. For information about how the subsequent terms and policies are evaluated, see "How a Routing Policy Chain Is Evaluated" on page 23.

NOTE: If you specify a policy chain as a subroutine, the entire chain acts as a single subroutine. As with other chains, the action specified by the default policy is taken only when the entire chain does not accept or reject a route.

NOTE: If a term defines multiple match conditions, including a subroutine, and a route does not match a condition specified before the subroutine, the evaluation of the term ends and the subroutine is not called and evaluated. In this situation, an action specified in the subroutine that manipulates a route's characteristics is not implemented.

Figure 10: Routing Policy Subroutine Evaluation



Routing Policy Tests

After you have created a routing policy, you can use the **test policy** command to ensure that the policy produces the results that you expect before applying the policy in a live environment. This command determines whether the routes specified in your routing policy are accepted or rejected. The default action of the **test policy** command is **accept**.

NOTE: The default policy of the **test policy** command accepts all routes from all protocols. Test output can be misleading when you are evaluating protocol-specific conditions.

For example, if you define a policy for BGP that accepts routes of a specified prefix and apply it to BGP as an export policy, BGP routes that match the prefix are advertised to BGP peers. However, if you test the same policy using the **test policy** command, the test output might indicate that non-BGP routes have been accepted.

CHAPTER 3

Introduction to Extended Match Conditions Configuration

Overview of BGP Communities and Extended Communities as Routing Policy Match Conditions

A *BGP community* is a group of destinations that share a common property. Community information is included as a path attribute in BGP update messages. This information identifies community members and allows you to perform actions on a group without having to elaborate upon each member. You can create a named community and include it in a routing policy with the **community** match condition, which is described in "Configuring Match Conditions in Routing Policy Terms" on page 47. For a list of the actions that can be configured for communities, see "Configuring Actions That Manipulate Route Characteristics" on page 56.

You can configure the standard community attribute and the extended communities attribute for inclusion in BGP update messages. The standard community attribute is four octets whereas the extended communities attribute is eight octets, providing a larger range for grouping or categorizing communities. You can use community and extended communities attributes to trigger routing decisions, such as acceptance, rejection, preference, or redistribution.

The BGP community attribute format is ***as-number:community-value***. The BGP extended communities attribute format instead has three fields: ***type:administrator:assigned-number***.

When specifying community IDs for the standard community attribute, you can use UNIX-style regular expressions. Regular expressions are not supported for the extended communities attribute.

NOTE: You can assign community tags to non-BGP routes through configuration (for static, aggregate, or generated routes) or an import routing policy. These tags can then be matched when BGP exports the routes.

To use a BGP community or extended community as a routing policy match condition, you define the community and its members and then include the community in a match condition.

The Junos OS supports the following standard:

- RFC 1997, *BGP Communities Attribute*

For configuration instructions, see the following topics:

- Defining BGP Communities and Extended Communities for Use in Routing Policy Match Conditions on page 116
- Including BGP Communities and Extended Communities in Routing Policy Match Conditions on page 122
- How BGP Communities and Extended Communities Are Evaluated in Routing Policy Match Conditions on page 28
- Using Routing Policies to Prevent Advertisement of BGP Communities to Neighbors on page 123
- Examples: Configuring BGP Communities as Routing Policy Match Conditions on page 123

How BGP Communities and Extended Communities Are Evaluated in Routing Policy Match Conditions

When you use BGP communities and extended communities as match conditions in a routing policy, the policy framework software evaluates them as follows:

- Each route is evaluated against each named community in a routing policy **from** statement. If a route matches one of the named communities in the **from** statement, the evaluation of the current term continues. If a route does not match, the evaluation of the current term ends.
- The route is evaluated against each member of a named community. The evaluation of all members must be successful for the named community evaluation to be successful.
- Each member in a named community is identified by either a literal community value or a regular expression (for information about using regular expressions, see "Using UNIX Regular Expressions in Community Names" on page 118). Each member is evaluated against each community associated with the route. (Communities are an unordered property of a route. For example, **1:2 3:4** is the same as **3:4 1:2**.) Only one community from the route is required to match for the member evaluation to be successful.
- Community regular expressions are evaluated on a character-by-character basis. For example, if a route contains community **1234:5678**, the regular expressions see nine discrete characters, including the colon (:), instead of two sets of numbers (1234 and 5678) separated by a colon. For example:

  [edit]

```
policy-options {
  policy-statement one {
    from {
      community [comm-one comm-two];
    }
  }
  community comm-one members [ 1:2 "^4:(5|6)$" ];
  community comm-two members [ 7:8 9:10 ];
}
```

If a community member is a regular expression, a string match is made rather than a numeric match.

For example:

```
community example1 members 100:100
community example2 members 100:1..
```

Given a route with a community value of 1100:100, this route matches **community example2** but not **example1**.

- To match routing policy **one**, the route must match either **comm-one** or **comm-two**.
- To match **comm-one**, the route must have a community that matches **1:2** and a community that matches **4:5** or **4:6**.
- To match **comm-two**, the route must have a community that matches **7:8** and a community that matches **9:10**.

Multiple Matches

When multiple matches are found, label aggregation does not happen. Consider the following configuration:

```
family inet-vpn {
  unicast {
    aggregate-label {
      community community-name;
    }
  }
}

family inet-vpn {
  labeled-unicast {
    aggregate-label {
      community community-name;
    }
  }
}
```

Suppose, for instance, that two routes are received with community attributes **target:65000:1000 origin:65200:2000** and that the community name is **"5...:.*"**. In this case, both the extended community attributes, **target:65000:1000** and **origin:65200:2000** match the regular expression of the community name. In this case, label aggregation does not occur. In the following example, the **Label operation** field shows that the labels are not aggregated.

```
user@host> show route table VPN detail | match "^10 | Communities | Push"
10.1.1.0/30 (1 entry, 1 announced)
                Label operation: Push 101040
                Push 101040
                Communities: target:65000:1000 origin:65200:2000
10.1.1.4/30 (1 entry, 1 announced)
                Label operation: Push 101056
                Push 101056
                Communities: target:65000:1000 origin:65200:2000
```

You can resolve this issue in either of the following ways:

- Be more specific in the regular expression if the site-of-origin extended community attribute does not overlap with the target one.
- Specify the site of origin in the community name.

Both methods are shown in the following examples.

Be More Specific in the Regular Expression

```
user@host# set policy-options community community-name members "52...:.*"
user@host# commit
```

```
user@host> show route table VPN detail | match "^10 | Communities | Push"
10.1.1.0/30 (1 entry, 1 announced)
                Label operation: Push 101040
                Push 101040
                Communities: target:65000:1000 origin:65200:2000
10.1.1.4/30 (1 entry, 1 announced)
                Label operation: Push 101040
                Push 101040
                Communities: target:65000:1000 origin:65200:2000
```

Specify the Site of Origin in the Community Name

```
user@host# set policy-options community community-name members "origin:65...:.*"
user@host# commit
```

```
user@host> show route table VPN detail | match "^10 | Communities | Push"
10.1.1.0/30 (1 entry, 1 announced)
                Label operation: Push 101040
                Push 101040
                Communities: target:65000:1000 origin:65200:2000
10.1.1.4/30 (1 entry, 1 announced)
                Label operation: Push 101040
                Push 101040
                Communities: target:65000:1000 origin:65200:2000
```

Extended Community Type

The extended community type is not taken into account by regular expressions. Consider, for instance, the following community attributes and community name.

Communities:

- 5200:1000
- **target:65000:1000**
- **origin:65200:2000**

Community attribute:

- community-name members "5...:.*"

In this case, both extended community attribute, **5200:1000** and the extended community attribute, **origin:65200:2000**, match the regular expression of the community name. Therefore, the label aggregation does not occur, as shown here:

```
user@host> show route table VPN detail | match "^10 | Communities | Push"
10.1.1.0/30 (1 entry, 1 announced)
                Label operation: Push 101040
                Push 101040
                Communities: 5200:1000 target:65000:1000 origin:65200:2000
10.1.1.4/30 (1 entry, 1 announced)
                Label operation: Push 101056
                Push 101056
                Communities: 5200:1000 target:65000:1000 origin:65200:2000
```

You can resolve this issue by using a more specific regular expression. For example, you can use the anchor character (^) to bind the location of the digits, as shown here:

```
user@host# set policy-options community community-name members "^5...:.*"
user@host# commit
```

```
user@host> show route table VPN detail | match "^10 | Communities | Push"
10.1.1.0/30 (1 entry, 1 announced)
                Label operation: Push 101040
                Push 101040
                Communities: 5200:1000 target:65000:1000 origin:65200:2000
10.1.1.4/30 (1 entry, 1 announced)
                Label operation: Push 101040
                Push 101040
                Communities: 5200:1000 target:65000:1000 origin:65200:2000
```

Multiple Communities Are Matched with Ex-OR Logic

This differs from the AND matching logic used for non-extended communities in BGP.

If, for instance, four routes are received with two sets of community attributes, the regular expression might match both community attributes. Consider the following example:

- Communities—5200:1000 target:65000:1000
- Communities—target:65000:1000 origin:65200:2000
- Community attribute—community community-name member ["^5...:.*" origin:65.*:.*]

Both labels are aggregated, as shown here:

```
user@host> show route table VPN detail | match "^10 | Communities | Push"
10.1.1.0/30 (1 entry, 1 announced)
                Label operation: Push 101040
                Push 101040
                Communities: target:65000:1000 origin:65200:2000
10.1.1.4/30 (1 entry, 1 announced)
                Label operation: Push 101040
                Push 101040
                Communities: target:65000:1000 origin:65200:2000
```

```
10.1.1.16/30 (1 entry, 1 announced)
                Label operation: Push 121104
                Push 101104
                Communities: 5200:1000 target:65000:1000
10.1.1.20/30 (1 entry, 1 announced)
                Label operation: Push 121104
                Push 101104
                Communities: 5200:1000 target:65000:1000
```

A more complete example of community values is shown here:

```
user@host> show policy-options community community-name
members [ "(^1...:*)|(^3...:*)|(^4...:*)" origin:2.*:* origin:3.*:* origin:6.*:*
 ]
```

This regular expression matches community values starting with 1, 3, or 4, and matches extended community values of type origin whose administrative value starts with 2, 3, or 6.

CHAPTER 4

Introduction to Extended Actions Configuration

Overview of Per-Packet Load Balancing

By default, when there are multiple equal-cost paths to the same destination for the active route, the Junos OS uses a hash algorithm to choose one of the next-hop addresses to install in the forwarding table. Whenever the set of next hops for a destination changes in any way, the next-hop address is rechosen using the hash algorithm.

You can configure the Junos OS so that, for the active route, all next-hop addresses for a destination are installed in the forwarding table. This feature is called per-packet load balancing. You can use load balancing to spread traffic across multiple paths between routers. The behavior of the load-balance per-packet function depends on the version of the Internet Processor application-specific integrated circuit (ASIC) in your routing platform:

- On routing platforms with the Internet Processor ASIC, when per-packet load balancing is configured, traffic between routers with multiple paths is spread using the hash algorithm across the available interfaces. The forwarding table balances the traffic headed to a destination, transmitting it in round-robin fashion among the multiple next hops (up to a maximum of eight equal-cost load-balanced paths). The traffic is load-balanced on a per-packet basis.
- On routing platforms with the Internet Processor II ASIC, when per-packet load balancing is configured, traffic between routers with multiple paths is divided into individual traffic flows (up to a maximum of 16 equal-cost load-balanced paths). Packets for each individual flow are kept on a single interface.

Related Documentation

- Configuring Per-Packet Load Balancing on page 160
- Configuring Load Balancing Based on MPLS Labels on page 163
- Configuring Load Balancing for Ethernet Pseudowires on page 166
- Configuring Load Balancing Based on MAC Addresses on page 167
- Configuring VPLS Load Balancing Based on IP and MPLS Information on page 168

- Configuring VPLS Load Balancing on MX Series 3D Universal Edge Routers on page 169

CHAPTER 5

Introduction to Traffic Sampling, Forwarding, and Monitoring Overview

Traffic Sampling, Forwarding, and Monitoring Overview

Traffic sampling allows you to sample IP traffic based on particular input interfaces and various fields in the packet header. You can also use traffic sampling to monitor any combination of specific logical interfaces, specific protocols on one or more interfaces, a range of addresses on a logical interface, or individual IP addresses. Information about the sampled packets is saved to files on the router's hard disk.

The forwarding policies allow you to configure the per-flow load balancing, port mirroring, and Domain Name System (DNS) or Trivial File Transfer Protocol (TFTP) forwarding. In Junos OS Release 9.0 and later, you can configure per-prefix load balancing. This feature enables the router to elect the next hop independent of the route chosen by other routers. The result is a better utilization of available links.

Traffic sampling and forwarding are supported only on routers equipped with an Internet Processor II application-specific integrated circuit (ASIC). To determine whether a routing platform has an Internet Processor II ASIC, use the **show chassis hardware** command.

Traffic sampling is not meant to capture all packets received by a router. We do not recommend excessive sampling (a rate greater than 1/1000 packets), because it can increase the load on your processor. If you need to set a higher sampling rate to diagnose a particular problem or type of traffic received, we recommend that you revert to a lower sampling rate after you discover the problem or troublesome traffic.

CHAPTER 6

Introduction to Traffic Forwarding and Monitoring Configuration

Per-Flow and Per-Prefix Load Balancing Overview

By default, when there are multiple equal-cost paths to the same destination, the Junos OS chooses one of the next-hop addresses at random.

On all M Series Multiservice Edge Routers, MX Series 3D Universal Edge Routers, and T Series Core Routers, you have the addition option of configuring per-prefix load balancing based on a specified hash value that enables the router to elect a next hop independently of the route chosen by other routers.

On the M120, M320, and MX Series routers only, you have the additional option of enabling per-flow load balancing based on a unique, load-balance hash value for each Packet Forwarding Engine slot.

Related Documentation

- Configuring Per-Prefix Load Balancing on page 209
- Configuring Per-Flow Load Balancing Based on Hash Values on page 210

PART 2

Configuration

CHAPTER 7

Routing Policy Configuration Statements

Configuring a Routing Policy

To create a routing policy, you can include the **policy-options** statement in the configuration:

```
policy-options {
   as-path name regular-expression;
   as-path-group group-name;
   community name {
      invert-match;
      members [ community-ids ];
   }
   condition condition-name {
      if-route-exists address table table-name;
   }
   damping name {
      disable;
      half-life minutes;
      max-suppress minutes;
      reuse number;
      suppress number;
   }
   policy-statement policy-name {
      term term-name {
         from {
            family;
            match-conditions;
            policy subroutine-policy-name;
            prefix-list name;
            route-filter destination-prefix match-type <actions>;
            source-address-filter source-prefix match-type <actions>;
         }
         to {
            match-conditions;
            policy subroutine-policy-name;
```

```
        }
        then actions;
        default-action (accept | reject);
      }
    }
    prefix-list name {
      ip-addresses;
    }
  }
  protocols {
    protocol-name {
      export [ policy-names ];
      import [ policy-names ];
    }
  }
```

Minimum Routing Policy Configuration

To define and apply a routing policy, you must include at least the following statements at the **[edit policy-options]** and **[edit protocols]** hierarchy levels. At the **[edit protocols]** hierarchy level, you can define one or more policy names.

```
[edit]
policy-options {
  policy-statement policy-name {
    term term-name {
      from {
        family family-name;
        match-conditions;
        prefix-list name;
        route-filter destination-prefix match-type <actions>;
        source-address-filter source-prefix match-type <actions>;
      }
      to {
        match-conditions;
      }
      then actions;
    }
  }
}
protocols {
  protocol-name {
    import [ policy-names ];
    export [ policy-names ];
  }
}
```

Minimum Routing Policy Chain Configuration

To define and apply a routing policy chain, you must include at least the following statements at the **[edit policy-options]** and **[edit protocols]** hierarchy levels. At the **[edit protocols]** hierarchy level, you can define a chain of policy names that are evaluated in order.

```
[edit]
policy-options {
  policy-statement policy-name {
    term term-name {
      from {
        family family-name;
        match-conditions;
        prefix-list name;
        route-filter destination-prefix match-type <actions>;
        source-address-filter source-prefix match-type <actions>;
      }
      to {
        match-conditions;
      }
      then actions;
    }
  }
  policy-statement policy-name {
    term term-name {
      from {
        family family-name;
        match-conditions;
        prefix-list name;
        route-filter destination-prefix match-type <actions>;
        source-address-filter source-prefix match-type <actions>;
      }
      to {
        match-conditions;
      }
      then actions;
    }
  }
  prefix-list name {
    ip-addresses;
  }
}
protocols {
  protocol-name {
    import [ policy-names ];
    export [ policy-names ];
  }
}
```

Minimum Subroutine Configuration

To configure a routing policy that calls a subroutine from another routing policy, you must include at least the following statements at the **[edit policy-options]** and **[edit protocols]** hierarchy levels. At the **[edit protocols *protocol-name* (export | import)]** hierarchy levels, you can specify one or more policy names.

```
[edit]
policy-options {
  policy-statement subroutine-policy-name {
    term term-name {
      from {
```

```
            family family-name;
            match-conditions;
            prefix-list name;
            route-filter destination-prefix match-type <actions>;
            source-address-filter source-prefix match-type <actions>;
          }
          to {
            match-conditions;
          }
          then actions;
        }
      }
      policy-statement policy-name {
        term term-name {
          from {
            family family-name;
            policy subroutine-policy-name;
          }
          to {
            policy subroutine-policy-name;
          }
          then actions;
        }
      }
    }
    protocols {
      protocol-name {
        import [ policy-names ];
        export [ policy-names ];
      }
    }
```

CHAPTER 8

Routing Policy Configuration

Defining Routing Policies

To define a routing policy, include the **policy-statement** statement:

```
policy-statement policy-name {
  term term-name {
    from {
      family family-name
      match-conditions;
      policy subroutine-policy-name;
      prefix-list name;
      route-filter destination-prefix match-type <actions>;
      source-address-filter source-prefix match-type <actions>;
    }
    to {
      match-conditions;
      policy subroutine-policy-name;
    }
    then actions;
  }
}
```

You can include this statement at the following hierarchy levels:

- **[edit policy-options]**
- **[edit logical-systems *logical-system-name* policy-options]**

policy-name specifies the policy name and must be unique in the configuration. It can contain letters, numbers, and hyphens (-) and can be up to 255 characters long. To include spaces in the name, enclose the entire name in quotation marks (“ ”).

term-name specifies the name of a term in the policy and must be unique in that policy. It can contain letters, numbers, and hyphens (-) and can be up to 64 characters long. To include spaces in the name, enclose the entire name in quotation marks (“ ”).

A policy statement can include multiple terms, including a unnamed term which must be the final term in the policy. To configure an unnamed term, omit the **term** statement when defining match conditions and actions. However, we recommend that you name all terms.

To list the routing policies by policy-statement ***policy-name*** in alphabetical order, enter the **show policy-options** configuration command.

For information about configuring the components of a term, see the following sections:

- Configuring Match Conditions in Routing Policy Terms on page 47

- Configuring Actions in Routing Policy Terms on page 54

Configuring Match Conditions in Routing Policy Terms

Each term in a routing policy can include two statements, **from** and **to**, to define the conditions that a route must match for the policy to apply:

```
from {
    family family-name;
    match-conditions;
    policy subroutine-policy-name;
    prefix-list name;
    route-filter destination-prefix match-type <actions>;
    source-address-filter source-prefix match-type <actions>;
}
to {
    match-conditions;
    policy subroutine-policy-name;
}
```

You can include these statements at the following hierarchy levels:

- **[edit policy-options** policy-statement ***policy-name*** **term** ***term-name*****]**
- **[edit logical-systems** ***logical-system-name*** **policy-options** policy-statement ***policy-name*** **term** ***term-name*****]**

In the **from** statement, you define the criteria that an incoming route must match. You can specify one or more match conditions. If you specify more than one, they all must match the route for a match to occur.

The **from** statement is optional. If you omit the **from** and the **to** statements, all routes are considered to match.

NOTE: In export policies, omitting the from statement from a routing policy term might lead to unexpected results. For more information, see "Applying Routing Policies and Policy Chains to Routing Protocols" on page 64.

In the **to** statement, you define the criteria that an outgoing route must match. You can specify one or more match conditions. If you specify more than one, they all must match the route for a match to occur. You can specify most of the same match conditions in the **to** statement that you can in the **from** statement. In most cases, specifying a match condition in the **to** statement produces the same result as specifying the same match condition in the **from** statement.

The **to** statement is optional. If you omit both the **to** and the **from** statements, all routes are considered to match.

NOTE: All conditions in the **from** and **to** statements must match for the action to be taken. The match conditions defined in Table 5 on page 48 are effectively a logical AND operation. Matching in prefix lists and route lists is handled differently. They are effectively a logical OR operation. For more information about how matching occurs for prefix lists and route lists, including how they are evaluated, see "Configuring Prefix Lists for Use in Routing Policy Match Conditions" on page 127 and "Configuring Route Lists for Use in Routing Policy Match Conditions" on page 131. If you configure a policy that includes some combination of route filters, prefix lists, and source address filters, they are evaluated according to a logical OR operation or a longest-route match lookup.

Table 5 on page 48 describes the match conditions available for matching an incoming or outgoing route. The table indicates whether you can use the match condition in both **from** and **to** statements and whether the match condition functions the same or differently when used with both statements. If a match condition functions differently in a **from** statement than in a **to** statement, or if the condition cannot be used in one type of statement, there is a separate description for each type of statement. Otherwise, the same description applies to both types of statements.

Table 5 on page 48 also indicates whether the match condition is standard or extended. In general, the extended match conditions include criteria that are defined separately from the routing policy (autonomous system [AS] path regular expressions, communities, and prefix lists) and are more complex than standard match conditions. The extended match conditions provide many powerful capabilities. The standard match conditions include criteria that are defined within a routing policy and are less complex than the extended match conditions.

Table 5: Routing Policy Match Conditions

Match Condition	Match Condition Category	from Statement Description	to Statement Description
aggregate-contributor	Standard	Match routes that are contributing to a configured aggregate. This match condition can be used to suppress a contributor in an aggregate route.	
area *area-id*	Standard	(Open Shortest Path First [OSPF] only) Area identifier. In a **from** statement used with an **export** policy, match a route learned from the specified OSPF area when exporting OSPF routes into other protocols.	
as-path *name*	Extended	(Border Gateway Protocol [BGP] only) Name of an AS path regular expression. For more information, see "Configuring AS Path Regular Expressions to Use as Routing Policy Match Conditions" on page 109.	
as-path-group *group-name*	Extended	(BGP only) Name of an AS path group regular expression. For more information, see "Configuring AS Path Regular Expressions to Use as Routing Policy Match Conditions" on page 109.	

Table 5: Routing Policy Match Conditions *(continued)*

Match Condition	Match Condition Category	from Statement Description	to Statement Description
color *preference* **color2** *preference*	Standard	Color value. You can specify preference values (**color** and **color2**) that are finer-grained than those specified in the **preference** and **preference2** match conditions. The color value can be a number in the range from 0 through 4,294,967,295 ($2^{32} - 1$). A lower number indicates a more preferred route.	
community-count *value* **(equal | orhigher | orlower)**	Standard	(BGP only) Number of community entries required for a route to match. The count value can be a number in the range of 0 through 1,024. Specify one of the following options: • **equal**—The number of communities must equal this value to be considered a match. • **orhigher** —The number of communities must be greater than or equal to this value to be considered a match. • **orlower**—The number of communities must be less than or equal to this value to be considered a match. NOTE: If you configure multiple **community-count** statements, the matching is effectively a logical AND operation. NOTE: The **community-count** attribute only works with standard communities. It does not work with extended communities.	You cannot specify this match condition.
community [*names* **]**	Extended	Name of one or more communities. If you list more than one name, only one name needs to match for a match to occur (the matching is effectively a logical OR operation). For more information, see "Overview of BGP Communities and Extended Communities as Routing Policy Match Conditions" on page 27.	
external [type *metric-type*]	Standard	(OSPF and IS-IS only) Match IGP external routes. For IS-IS routes, the **external** condition also matches routes that are exported from one IS-IS level to another. The **type** keyword is optional and is applicable only to OSPF external routes. When you do not specify **type**, the **external** condition matches all IGP external (OSPF and IS-IS) routes. When you specify **type**, the **external** condition matches only OSPF external routes with the specified OSPF metric type. The metric type can either be 1 or 2. To match BGP external routes, use the **route-type** match condition.	
family *family-name*	Standard	Name of an address family. ***family-name*** can be either **inet** or **inet6**. Match the address family IP version 4 (IPv4) or IP version 6 (IPv6) of the route. Default setting is **inet**.	
instance *instance-name*	Standard	Name of one or more routing instances. Match a route learned from one of the specified instances.	Name of one or more routing instances. Match a route to be advertised over one of the specified instances.

Table 5: Routing Policy Match Conditions *(continued)*

Match Condition	Match Condition Category	from Statement Description	to Statement Description
interface *interface-name*	Standard	Name or IP address of one or more routing device interfaces. Do not use this qualifier with protocols that are not interface-specific, such as IBGP. Match a route learned from one of the specified interfaces. Direct routes match routes configured on the specified interface.	Name or IP address of one or more routing device interfaces. Do not use this qualifier with protocols that are not interface-specific, such as IBGP. Match a route to be advertised from one of the specified interfaces.
level *level*	Standard	(Intermediate System-to-Intermediate System [IS-IS] only) IS-IS level. Match a route learned from a specified level.	(IS-IS only) IS-IS level. Match a route to be advertised to a specified level.
local-preference *value*	Standard	(BGP only) BGP local preference (**LOCAL_PREF**) attribute. The preference value can be a number in the range 0 through 4,294,967,295 ($2^{32} - 1$).	
metric *metric* **metric2** *metric* **metric3** *metric* **metric4** *metric*	Standard	Metric value. You can specify up to four metric values, starting with **metric** (for the first metric value) and continuing with **metric2**, **metric3**, and **metric4**. (BGP only) **metric** corresponds to the multiple exit discriminator (MED), and **metric2** corresponds to the interior gateway protocol (IGP) metric if the BGP next hop runs back through another route.	
multicast-scoping (*scoping-name* | *number*) < (**orhigher** | **orlower**) >	Standard	Multicast scope value of IPv4 or IPv6 multicast group address. The multicast-scoping name corresponds to an IPv4 prefix. You can match on a specific multicast-scoping prefix or on a range of prefixes. Specify **orhigher** to match on a scope and numerically higher scopes, or **orlower** to match on a scope and numerically lower scopes. For more information, see the *Junos OS Multicast Protocols Configuration Guide*. You can apply this scoping policy to the routing table by including the **scope-policy** statement at the **[edit routing-options]** hierarchy level. The ***number*** value can be any hexadecimal number from 0 through F. The multicast-scope value is a number from 0 through 15, or one of the following keywords with the associated meanings: • **node-local** (value=**1**)—No corresponding prefix • **link-local** (value=**2**)—Corresponding prefix **224.0.0.0/24** • **site-local** (value=**5**)—No corresponding prefix • **global** (value=**14**)—Corresponding prefix **224.0.1.0** through **238.255.255.255** • **organization-local** (value=**8**)—Corresponding prefix **239.192.0.0/14**	

Table 5: Routing Policy Match Conditions *(continued)*

Match Condition	Match Condition Category	from Statement Description	to Statement Description
neighbor *address*	Standard	Address of one or more neighbors (peers). For BGP, the address can be a directly connected or indirectly connected peer. For all other protocols, the address is the neighbor from which the advertisement is received. NOTE: The **neighbor *address*** match condition is not valid for the Routing Information Protocol (RIP).	Address of one or more neighbors (peers). For BGP import policies, specifying **to neighbor** produces the same result as specifying **from neighbor**. For BGP export policies, specifying the **neighbor** match condition has no effect and is ignored. For all other protocols, the **to** statement matches the neighbor to which the advertisement is sent. NOTE: The **neighbor *address*** match condition is not valid for the Routing Information Protocol (RIP).
next-hop *address*	Standard	Next-hop address or addresses specified in the routing information for a particular route. For BGP routes, matches are performed against each protocol next hop. NOTE: If you include a netmask with the next-hop address, the netmask is ignored and a system log message is generated. For more information about system log messages, see the *Junos OS System Log Messages Reference*.	
next-hop-type merged	Standard	LDP generates a next hop based on RSVP and IP next hops available to use, combined with forwarding-class mapping.	You cannot specify this match condition.
origin *value*	Standard	(BGP only) BGP origin attribute, which is the origin of the AS path information. The value can be one of the following: • **egp**—Path information originated in another AS. • **igp**—Path information originated within the local AS. • **incomplete**—Path information was learned by some other means.	
policy [*policy-name*]	Extended	Name of a policy to evaluate as a subroutine. For information about this extended match condition, see "Configuring Subroutines in Routing Policy Match Conditions" on page 146.	
preference *preference* **preference2 *preference***	Standard	Preference value. You can specify a primary preference value (**preference**) and a secondary preference value (**preference2**). The preference value can be a number from 0 through 4,294,967,295 ($2^{32} - 1$). A lower number indicates a more preferred route. To specify even finer-grained preference values, see the **color** and **color2** match conditions in this table.	

Table 5: Routing Policy Match Conditions *(continued)*

Match Condition	Match Condition Category	from Statement Description	to Statement Description
prefix-list ***prefix-list-name ip-addresses***	Extended	Named list of IP addresses. You can specify an exact match with incoming routes. For information about this extended match condition, see "Configuring Prefix Lists for Use in Routing Policy Match Conditions" on page 127.	You cannot specify this match condition.
prefix-list-filter ***prefix-list-name match-type***	Extended	Named prefix list. You can specify prefix length qualifiers for the list of prefixes in the prefix list. For information about this extended match condition, see "Configuring Prefix Lists for Use in Routing Policy Match Conditions" on page 127.	You cannot specify this match condition.
protocol ***protocol***	Standard	Name of the protocol from which the route was learned or to which the route is being advertised. It can be one of the following: **access**, **access-internal**, **aggregate**, **bgp**, **direct**, **dvmrp**, **isis**, **local**, **ospf**, **ospf2**, **ospf3**, **pim-dense**, **pim-sparse**, **rip**, **ripng**, or **static**. NOTE: The **ospf2** statement matches on OSPFv2 routes. The **ospf3** statement matches on OSPFv3 routes. The **ospf** statement matches on both OSPFv2 and OSPFv3 routes. For more information about access routes and access-internal routes, see "Example: Importing and Exporting Access and Access-Internal Routes in a Routing Policy" on page 88.	
rib ***routing-table***	Standard	Name of a routing table. The value of ***routing-table*** can be one of the following: • **inet.0**—Unicast IPv4 routes • ***instance-name*** **inet.0**—Unicast IPv4 routes for a particular routing instance • **inet.1**—Multicast IPv4 routes • **inet.2**—Unicast IPv4 routes for multicast reverse-path forwarding (RPF) lookup • **inet.3**—MPLS routes • **mpls.0**—MPLS routes for label-switched path (LSP) next hops • **inet6.0**—Unicast IPv6 routes	
route-filter ***destination-prefix match-type <actions>***	Extended	List of destination prefixes. When specifying a destination prefix, you can specify an exact match with a specific route or a less precise match using match types. You can configure either a common action that applies to the entire list or an action associated with each prefix. For more information, see "Configuring Route Lists for Use in Routing Policy Match Conditions" on page 131.	You cannot specify this match condition.

Table 5: Routing Policy Match Conditions *(continued)*

Match Condition	Match Condition Category	from Statement Description	to Statement Description
route-type ***value***	Standard	Type of BGP route. The value can be one of the following: • **external**—External route. • **internal**—Internal route. To match IGP external routes, use the **external** match condition.	
source-address-filter ***destination-prefix match-type <actions>***	Extended	List of multicast source addresses. When specifying a source address, you can specify an exact match with a specific route or a less precise match using match types. You can configure either a common action that applies to the entire list or an action associated with each prefix. For more information, see "Configuring Route Lists for Use in Routing Policy Match Conditions" on page 131.	You cannot specify this match condition.
state (active | inactive)	Standard	(BGP export only) Match on the following types of advertised routes: • **active**—An active BGP route • **inactive**—A route advertised to internal BGP peers as the best external path even if the best path is an internal route • **inactive**—A route advertised by BGP as the best route even if the routing table did not select it to be an active route	
tag ***string*** **tag2** ***string***	Standard	Tag value. You can specify two tag strings: **tag** (for the first string) and **tag2**. These values are local to the router and can be set on configured routes or by using an import routing policy. You can specify multiple tags under one match condition by including the tags within a bracketed list. For example: **from tag [tag1 tag2 tag3];** For OSPF routes, the**tag** action sets the 32-bit tag field in OSPF external link-state advertisement (LSA) packets. For IS-IS routes, the **tag** action sets the 32-bit flag in the IS-IS IP prefix type length values. (TLV). OSPF stores the INTERNAL route's OSPF area ID in the**tag2** attribute. However, for EXTERNAL routes, OSPF does not store anything in the **tag2**attribute. You can configure a policy term to set the **tag2** value for a route. If the route, already has a **tag2** value (for example, an OSPF route that stores area id in tag2), then the original **tag2** value is overwritten by the new value. When the policy contains the "from area" match condition, for internal OSPF routes, where **tag2** is set, based on the OSPF area- ID, the evaluation is conducted to compare the **tag2** attribute with the area ID. For external OSPF routes that do not have the **tag2** attribute set, the match condition fails.	

Related Documentation

- Routing Policy Examples on page 77

Configuring Actions in Routing Policy Terms

Each term in a routing policy can include a **then** statement, which defines the actions to take if a route matches all the conditions in the **from** and **to** statements in the term:

```
then {
  actions;
}
```

You can include this statement at the following hierarchy levels:

- **[edit policy-options** policy-statement ***policy-name*** **term** ***term-name*****]**
- **[edit logical-systems** ***logical-system-name*** **policy-options** policy-statement ***policy-name*** **term** ***term-name*****]**

If a term does not have **from** and **to** statements, all routes are considered to match, and the actions apply to them all. For information about the **from** and **to** statements, see "Configuring Match Conditions in Routing Policy Terms" on page 47.

You can specify one or more actions in the **then** statement. There are three types of actions:

- Flow control actions, which affect whether to accept or reject the route and whether to evaluate the next term or routing policy.
- Actions that manipulate route characteristics.
- Trace action, which logs route matches.

NOTE: When you specify an action that manipulates the route characteristics, the changes occur in a copy of the source route. The source route itself does not change. The effect of the action is visible only after the route is imported into or exported from the routing table. To view the source route before the routing policy has been applied, use the **show route receive-protocol** command. To view a route after an export policy has been applied, use the **show route advertised-protocol** command.

During policy evaluation, the characteristics in the copy of the source route always change immediately after the action is evaluated. However, the route is not copied to the routing table or a routing protocol until the completion of the policy evaluation is complete.

The **then** statement is optional. If you omit it, one of the following occurs:

- The next term in the routing policy, if one is present, is evaluated.
- If there are no more terms in the routing policy, the next routing policy, if one is present, is evaluated.

- If there are no more terms or routing policies, the accept or reject action specified by the default policy is taken. For more information, see "Default Routing Policies and Actions" on page 16.

The following sections discuss the following actions:

Configuring Flow Control Actions

Table 6 on page 55 lists the flow control actions. You can specify one of these actions along with the trace action or one or more of the actions that manipulate route characteristics (see "Configuring Actions That Manipulate Route Characteristics" on page 56).

Table 6: Flow Control Actions

Flow Control Action	Description
accept	Accept the route and propagate it. After a route is accepted, no other terms in the routing policy and no other routing policies are evaluated.
default-action accept	Accept and override any action intrinsic to the protocol. This is a nonterminating policy action.
reject	Reject the route and do not propagate it. After a route is rejected, no other terms in the routing policy and no other routing policies are evaluated.
default-action reject	Reject and override any action intrinsic to the protocol. This is a nonterminating policy action.
next term	Skip to and evaluate the next term in the same routing policy. Any accept or reject action specified in the **then** statement is skipped. Any actions in the **then** statement that manipulate route characteristics are applied to the route. **next term** is the default control action if a match occurs and you do not specify a flow control action.
next policy	Skip to and evaluate the next routing policy. Any accept or reject action specified in the **then** statement is skipped. Any actions in the **then** statement that manipulate route characteristics are applied to the route. **next policy** is the default control action if a match occurs, you do not specify a flow control action, and there are no further terms in the current routing policy.

Configuring Actions That Manipulate Route Characteristics

You can specify one or more of the actions listed in Table 7 on page 56 to manipulate route characteristics.

Table 7: Actions That Manipulate Route Characteristics

Action	Description
as-path-prepend *as-path*	(BGP only) Affix one or more AS numbers at the beginning of the AS path. If specifying more than one AS number, enclose the numbers in quotation marks (" "). The AS numbers are added after the local AS number has been added to the path. This action adds AS numbers to AS sequences only, not to AS sets. If the existing AS path begins with a confederation sequence or set, the affixed AS numbers are placed with a nonconfederation sequence. Otherwise, the affixed AS numbers are placed with a nonconfederation sequence. For more information, see "Prepending AS Numbers to BGP AS Paths" on page 153. In Junos OS Release 9.1 and later, you can specify 4-byte AS numbers as defined in RFC 4893, *BGP Support for Four-octet AS Number Space*, as well as the 2-byte AS numbers that are supported in earlier releases of the Junos OS.
as-path-expand last-as count *n*	(BGP only) Extract the last AS number in the existing AS path and affix that AS number to the beginning of the AS path ***n*** times, where ***n*** is a number from 1 through 32. The AS number is added before the local AS number has been added to the path. This action adds AS numbers to AS sequences only, not to AS sets. If the existing AS path begins with a confederation sequence or set, the affixed AS numbers are placed within a confederation sequence. Otherwise, the affixed AS numbers are placed within a nonconfederation sequence. This option is typically used in non-IBGP export policies.
class *class-name*	(Class of service [CoS] only) Apply the specified class-of-service parameters to routes installed into the routing table. For more information, see the *Junos OS Class of Service Configuration Guide*.
color *preference* **color2** *preference*	Set the preference value to the specified value. The **color** and **color2** preference values are even more fine-grained than those specified in the **preference** and **preference2** actions. The color value can be a number in the range from 0 through 4,294,967,295 ($2^{32} - 1$). A lower number indicates a more preferred route. If you set the preference with the **color** action, the value is internal to the Junos OS and is not transitive.
color (add | subtract) *number* **color2 (add | subtract)** *number*	Change the color preference value by the specified amount. If an addition operation results in a value that is greater than 4,294,967,295 ($2^{32} - 1$), the value is set to $2^{32} - 1$. If a subtraction operation results in a value less than 0, the value is set to 0. If an attribute value is not already set at the time of the addition or subtraction operation, the attribute value defaults to a value of 0 regardless of the amount specified. If you perform an addition to an attribute with a value of 0, the number you add becomes the resulting attribute value.
community (+ | add) [*names* **]**	(BGP only) Add the specified communities to the set of communities in the route. For more information, see "Overview of BGP Communities and Extended Communities as Routing Policy Match Conditions" on page 27.
community (– | delete) [*names* **]**	(BGP only) Delete the specified communities from the set of communities in the route. For more information, see "Overview of BGP Communities and Extended Communities as Routing Policy Match Conditions" on page 27.

Table 7: Actions That Manipulate Route Characteristics *(continued)*

Action	Description
community (= | set) [***names*** **]**	(BGP only) Replace any communities that were in the route in with the specified communities. For more information, see "Overview of BGP Communities and Extended Communities as Routing Policy Match Conditions" on page 27.
cos-next-hop-map ***map-name***	Set CoS-based next-hop map in forwarding table.
damping ***name***	(BGP only) Apply the specified route-damping parameters to the route. These parameters override the default damping parameters. This action is useful only in an import policy, because the damping parameters affect the state of routes in the routing table. To apply damping parameters, you must enable BGP flap damping as described in the *Junos OS Routing Protocols Configuration Guide*, and you must create a named list of parameters as described in "Using Routing Policies to Damp BGP Route Flapping" on page 155.
destination-class ***destination-class-name***	Maintain packet counts for a route passing through your network, based on the destination address in the packet. You can do the following: • Configure group destination prefixes by configuring a routing policy; see "Defining Routing Policies" on page 46 and "Routing Policy Examples" on page 77. • Apply that routing policy to the forwarding table with the corresponding destination class; see "Applying Routing Policies to the Forwarding Table" on page 69. • Enable packet counting on one or more interfaces by including the **destination-class-usage** statement at the **[edit interfaces** ***interface-name*** **unit** ***logical-unit-number*** **family inet accounting]** hierarchy level (see the *Junos OS Class of Service Configuration Guide*). See "Routing Policy Examples" on page 77. • View the output by using one of the following commands: **show interfaces destination-class (all |** ***destination-class-name logical-interface-name***), **show interfaces** ***interface-name*** **extensive**, or **show interfaces** ***interface-name*** **statistics** (see the *Junos OS Interfaces Command Reference*). • To configure a packet count based on the source address, use the **source-class** statement described in this table.
external type ***metric***	Set the external metric type for routes exported by OSPF. You must specify the keyword **type**.
forwarding-class ***forwarding-class-name***	Create the forwarding class that includes packets based on both the destination address and the source address in the packet. You can do the following: • Configure group prefixes by configuring a routing policy; see "Defining Routing Policies" on page 46 and "Routing Policy Examples" on page 77. • Apply that routing policy to the forwarding table with the corresponding forwarding class; see "Applying Routing Policies to the Forwarding Table" on page 69. • Enable packet counting on one or more interfaces by using the procedure described in either the **destination-class** or **source-class** actions defined in this table.
install-nexthop <strict> lsp ***lsp-name***	Choose which next hops, among a set of equal LSP next hops, are installed in the forwarding table. Use the export policy for the forwarding table to specify the LSP next hop to be used for the desired routes. Specify the **strict** option to enable strict mode, which checks to see if any of the LSP next hops specified in the policy are up. If none of the specified LSP next hops are up, the policy installs the discard next hop.

Table 7: Actions That Manipulate Route Characteristics *(continued)*

Action	Description
load-balance per-packet	(For export to the forwarding table only) Install all next-hop addresses in the forwarding table and have the forwarding table perform per-packet load balancing. This policy action allows you to optimize VPLS traffic flows across multiple paths. For more information, see "Configuring Per-Packet Load Balancing" on page 160.
local-preference *value*	(BGP only) Set the BGP local preference (**LOCAL_PREF**) attribute. The preference value can be a number in the range from 0 through 4,294,967,295 ($2^{32} - 1$).
local-preference (add | subtract) *number*	Change the local preference value by the specified amount. If an addition operation results in a value that is greater than 4,294,967,295 ($2^{32} - 1$), the value is set to $2^{32} - 1$. If a subtraction operation results in a value less than 0, the value is set to 0. If an attribute value is not already set at the time of the addition or subtraction operation, the attribute value defaults to a value of 0 regardless of the amount specified. If you perform an addition to an attribute with a value of 0, the number you add becomes the resulting attribute value. For BGP, if the attribute value is not known, it is initialized to 100 before the routing policy is applied.
map-to-interface (*interface-name* **| self)**	Sets the **map-to-interface** value which is similar to existing metric or tag actions. The **map-to-interface** action requires you to specify one of the following: • A logical interface (for example, ge-0/0/0.0). The logical interface can be any interface that multicast currently supports, including VLAN and aggregated Ethernet interfaces. NOTE: If you specify a physical interface as the **map-to-interface** (for example, ge-0/0/0), a value of **.0** is appended to physical interface to create a logical interface. • The keyword **self**. The **self** keyword specifies that multicast data packets are sent on the same interface as the control packets and no mapping occurs. If no term matches, then no multicast data packets are sent.
metric *metric* **metric2** *metric* **metric3** *metric* **metric4** *metric*	Set the metric. You can specify up to four metric values, starting with **metric** (for the first metric value) and continuing with **metric2**, **metric3**, and **metric4**. (BGP only) **metric** corresponds to the MED, and **metric2** corresponds to the IGP metric if the BGP next hop loops through another router.
metric (add | subtract) *number* **metric2 (add | subtract)** *number* **metric3 (add | subtract)** *number* **metric4 (add | subtract)** *number*	Change the metric value by the specified amount. If an addition operation results in a value that is greater than 4,294,967,295 ($2^{32} - 1$), the value is set to $2^{32} - 1$. If a subtraction operation results in a value less than 0, the value is set to 0. If an attribute value is not already set at the time of the addition or subtraction operation, the attribute value defaults to a value of 0 regardless of the amount specified. If you perform an addition to an attribute with a value of 0, the number you add becomes the resulting attribute value.
metric expression (metric multiplier *x* **offset** *a* **| metric2 multiplier** *y* **offset** *b*)	Calculate a metric based on the current values of **metric** and **metric2**. This policy action overrides the current value of the metric attribute with the result of the expression ((x * metric) + a) + ((y * metric2) + b) where **metric** and **metric2** are the current input values. Metric multipliers are limited in range to eight significant digits.

Table 7: Actions That Manipulate Route Characteristics *(continued)*

Action	Description
metric (igp | minimum-igp) ***site-offset***	(BGP only) Change the metric (MED) value by the specified negative or positive offset. This action is useful only in an external BGP (EBGP) export policy.

next-hop (*address* | discard | next-table *table-name* | peer-address | reject | self)	Set the next-hop address. When the advertising protocol is BGP, you can set the next hop only when any third-party next hop can be advertised; that is, when you are using IBGP or EBGP confederations.
	
	If you specify **self**, the next-hop address is replaced by one of the local routing device's addresses. The advertising protocol determines which address to use. When the advertising protocol is BGP, this address is set to the local IP address used for the BGP adjacency. A routing device cannot install routes with itself as the next hop.
	
	If you specify **peer-address**, the next-hop address is replaced by the peer's IP address. This option is valid only in import policies. Primarily used by BGP to enforce using the peer's IP address for advertised routes, this option is meaningful only when the next hop is the advertising routing device or another directly connected routing device.
	
	If you specify **discard**, the next-hop address is replaced by a discard next hop.
	
	If you specify **next-table**, the routing device performs a forwarding lookup in the specified table.
	
	If you use the **next-table** action, the configuration must include a term qualifier that specifies a different table than the one specified in the **next-table** action. In other words, the term qualifier in the **from** statement must exclude the table in the **next-table** action. In the following example, the first term contains **rib vrf-customer2.inet.0** as a matching condition. The action specifies a next-hop in a different routing table, **vrf-customer1.inet.0**. The second term does the opposite by using **rib vrf-customer1.inet.0** in the match condition and **vrf-customer2.inet.0** In the **next-table** action.
	
	`term 1 {`
	`  from {`
	`  protocol bgp;`
	`  rib vrf-customer2.inet.0;`
	`  community customer;`
	`  }`
	`  then {`
	`  next-hop next-table vrf-customer1.inet.0;`
	`  }`
	`}`
	`term 2 {`
	`  from {`
	`  protocol bgp;`
	`  rib vrf-customer1.inet.0;`
	`  community customer;`
	`  }`
	`  then {`
	`  next-hop next-table vrf-customer2.inet.0;`
	`  }`
	`}`
	
	If you specify **reject**, the next-hop address is replaced by a reject next hop.

Table 7: Actions That Manipulate Route Characteristics *(continued)*

Action	Description
origin *value*	(BGP only) Set the BGP origin attribute to one of the following values: • **igp**—Path information originated within the local AS. • **egp**—Path information originated in another AS. • **incomplete**—Path information learned by some other means.
preference *preference* **preference2** *preference*	Set the preference value. You can specify a primary preference value (**preference**) and a secondary preference value (**preference2**). The preference value can be a number in the range from 0 through 4,294,967,295 ($2^{32} - 1$). A lower number indicates a more preferred route. To specify even finer-grained preference values, see the **color** and **color2** actions in this table. If you set the preference with the **preference** action, the new preference remains associated with the route. The new preference is internal to the Junos OS and is not transitive.
preference (add | subtract) *number* **preference2 (add | subtract)** *number*	Change the preference value by the specified amount. If an addition operation results in a value that is greater than 4,294,967,295 ($2^{32} - 1$), the value is set to $2^{32} - 1$. If a subtraction operation results in a value less than 0, the value is set to 0. If an attribute value is not already set at the time of the addition or subtraction operation, the attribute value defaults to a value of 0 regardless of the amount specified. If you perform an addition to an attribute with a value of 0, the number you add becomes the resulting attribute value.
priority (low | medium | high)	(OSPF import only) Specify a priority for prefixes included in an OSPF import policy. Prefixes learned through OSPF are installed in the routing table based on the priority assigned to the prefixes. Prefixes assigned a priority of **high** are installed first, while prefixes assigned a priority of **low** are installed last. NOTE: An OSPF import policy can only be used to set priority or to filter OSPF external routes. If an OSPF import policy is applied that results in a **reject** terminating action for a nonexternal route, then the **reject** action is ignored and the route is accepted anyway.

Table 7: Actions That Manipulate Route Characteristics *(continued)*

Action	Description
source-class *source-class-name*	Maintain packet counts for a route passing through your network, based on the source address. You can do the following: • Configure group source prefixes by configuring a routing policy; see "Defining Routing Policies" on page 46 and "Routing Policy Examples" on page 77. • Apply that routing policy to the forwarding table with the corresponding source class; see "Applying Routing Policies to the Forwarding Table" on page 69. • Enable packet counting on one or more interfaces by including the **source-class-usage** *interface-name* statement at the **[edit interfaces** *logical-unit-number* **unit family inet accounting]** hierarchy level. Also, follow the **source-class-usage** statement with the **input** or **output** statement to define the inbound and outbound interfaces on which traffic monitored for source-class usage (SCU) is arriving and departing (or define one interface for both). The complete syntax is **[edit interfaces** *interface-name* **unit family inet accounting source-class-usage (input | output | input output)** *unit-number***]**. See the example in "Routing Policy Examples" on page 77. • View the output by using one of the following commands: **show interfaces** *interface-name* **source-class** *source-class-name*, **show interfaces** *interface-name* **extensive**, or **show interfaces** *interface-name* **statistics** (see the *Junos OS Interfaces Command Reference*). • To configure a packet count based on the destination address, use the **destination-class** statement described in this table. • For a detailed source-class usage example configuration, see the *Junos OS Source Class Usage Feature Guide*.
ssm-source [*addresses* **];**	Specify one or more IPv4 or IPv6 source addresses for the source-specific multicast (SSM) policy
ssm-source [*addresses* **];**	Specify one or more IPv4 or IPv6 source addresses for the source-specific multicast (SSM) policy.
tag *tag* **tag2** *tag*	Set the tag value. You can specify two tag strings: **tag** (for the first string) and **tag2** (a second string). These values are local to the router. • For OSPF routes the **tag** action sets the 32-bit tag field in OSPF external link-state advertisement (LSA) packets. • For IS-IS routes, the **tag** action sets the 32-bit flag in the IS-IS IP prefix type length values (TLV). • For RIPv2 routes, the **tag** action sets the route-tag community. The **tag2** option is not supported.
tag (add | subtract) *number* **tag2 (add | subtract)** *number*	Change the tag value by the specified amount. If an addition operation results in a value that is greater than 4,294,967,295 ($2^{32} - 1$), the value is set to $2^{32} - 1$. If a subtraction operation results in a value less than 0, the value is set to 0. If an attribute value is not already set at the time of the addition or subtraction operation, the attribute value defaults to a value of 0 regardless of the amount specified. If you perform an addition to an attribute with a value of 0, the number you add becomes the resulting attribute value.

Configuring the Default Action in Routing Policies

The **default-action** statement overrides any action intrinsic to the protocol. This action is also nonterminating, so that various policy terms can be evaluated before the policy is terminated. You can specify a default action, either **accept** or **reject**, as follows:

```
[edit]
policy-options {
  policy-statement policy-name {
    term term-name {
      from {
        family family-name;
        match-conditions;
        policy subroutine-policy-name;
        prefix-list name;
        route-filter destination-prefix match-type <actions>;
        source-address-filter source-prefix match-type <actions>;
      }
      to {
        match-conditions;
        policy subroutine-policy-name;
      }
      then {
        actions;
        default-action (accept | reject);
      }
    }
  }
}
```

The resulting action is set either by the protocol or by the last policy term that is matched.

Example: Configuring the Default Action in a Routing Policy

Configure a routing policy that matches routes based on three policy terms. If the route matches the first term, a certain community tag is attached. If the route matches two separate terms, then both community tags are attached. If the route does not match any terms, it is rejected (protocol's default action). Note that the terms **hub** and **spoke** are mutually exclusive.

```
[edit]
policy-options {
  policy-statement test {
    term set-default {
      then default-action reject;
    }
    term hub {
      from interface ge-2/1/0.5;
      then {
        community add test-01-hub;
        default-action accept;
      }
    }
    term spoke {
      from interface [ ge-2/1/0.1 ge-2/1/0.2 ];
      then {
        community add test-01-spoke;
        default-action accept;
      }
    }
    term management {
      from protocol direct;
```

```
        then {
          community add management;
          default-action accept;
        }
      }
    }
  }
```

Configuring a Final Action in Routing Policies

In addition to specifying an action using the **then** statement in a named term, you can also specify an action using the **then** statement in an unnamed term, as follows:

```
[edit]
policy-options {
  policy-statement policy-name {
    term term-name {
      from {
        family family-name;
        match-conditions;
        policy subroutine-policy-name;
        prefix-list name;
        route-filter destination-prefix match-type <actions>;
        source-address-filter source-prefix match-type <actions>;
      }
      to {
        match-conditions;
        policy subroutine-policy-name;
      }
      then {
        actions;
      }
    }
    then action;
  }
}
```

Logging Matches to a Routing Policy Term

If you specify the trace action, the match is logged to a trace file. To set up a trace file, you must specify the following elements in the global **traceoptions** statement:

- Trace filename
- **policy** option in the **flag** statement

The following example uses the trace filename of **policy-log**:

```
[edit]
routing-options {
  traceoptions {
    file "policy-log";
    flag policy;
  }
}
```

This action does not affect the flow control during routing policy evaluation.

If a term that specifies a trace action also specifies a flow control action, the name of the term is logged in the trace file. If a term specifies a trace action only, the word <**default**> is logged.

Configuring Separate Actions for Routes in Route Lists

If you specify route lists in the **from** statement, for each route in the list, you can specify an action to take on that individual route directly, without including a **then** statement. For more information, see "Configuring Route Lists for Use in Routing Policy Match Conditions" on page 131.

Applying Routing Policies and Policy Chains to Routing Protocols

For a routing policy to take effect, you must apply it to either a routing protocol or the forwarding table.

Before applying routing policies to routing protocols, you must know if each protocol supports import and export policies and the level at which you can apply these policies. "Protocols That Can Be Imported to and Exported from the Routing Table" on page 227 summarizes the import and export policy support for each routing protocol and the level at which you can apply these policies.

To apply one or more routing policies to a routing protocol, include the **import** and **export** statements:

```
import [ policy-names ];
export [ policy-names ];
```

You can include the statements at the following hierarchy levels:

- **[edit protocols *protocol-name*]**
- **[edit logical-systems *logical-system-name* protocols *protocol-name*]**

An ordered set of policies is referred to as a *policy chain*.

In the **import** statement, list the names of one or more routing policies to be evaluated when routes are imported into the routing table from the routing protocol.

In the **export** statement, list the names of one or more routing policies to be evaluated when routes are being exported from the routing table into a dynamic routing protocol. Only active routes are exported from the routing table.

You can reference the same routing policy one or more times in the same or different **import** and **export** statements.

The policy framework software evaluates the routing policies in a chain sequentially, from left to right. If an action specified in one of the policies manipulates a route characteristic, the policy framework software carries the new route characteristic forward during the evaluation of the remaining policies. For example, if the action specified in the first policy of a chain sets a route's metric to 500, this route matches the criterion of **metric 500** defined in the next policy.

For information about how the policy framework software evaluates routing policies and policy chains, see "Evaluating a Routing Policy" on page 22.

Applying Policy Expressions to Routes Exported from Routing Tables

Policy expressions give the policy framework software a different way to evaluate routing policies. A *policy expression* uses Boolean logical operators with policies. The logical operators establish rules by which the policies are evaluated.

During evaluation of a routing policy in a policy expression, the policy action of accept, reject, or next policy is converted to the value of TRUE or FALSE. This value is then evaluated against the specified logical operator to produce output of either TRUE or FALSE. The output is then converted back to a flow control action of accept, reject, or next policy. The result of the policy expression is applied as it would be applied to a single policy; the route is accepted or rejected and the evaluation ends, or the next policy is evaluated.

Table 8 on page 65 summarizes the policy actions and their corresponding TRUE and FALSE values and flow control action values. Table 9 on page 65 describes the logical operators. For complete information about policy expression evaluation, see "How a Policy Expression Is Evaluated" on page 67.

You must enclose a policy expression in parentheses. You can place a policy expression anywhere in the **import** or **export** statements and in the **from policy** statement.

Table 8: Policy Action Conversion Values

Policy Action	Conversion Value	Flow Control Action Conversion Value
Accept	TRUE	Accept
Reject	FALSE	Reject
Next policy	TRUE	Next policy

Table 9: Policy Expression Logical Operators

Logical Operator	Policy Expression Logic	How Logical Operator Affects Policy Expression Evaluation
&& (Logical AND)	Logical AND requires that all values must be TRUE to produce output of TRUE. Routing policy value of TRUE and TRUE produces output of TRUE. Value of TRUE and FALSE produces output of FALSE. Value of FALSE and FALSE produces output of FALSE.	If the first routing policy returns the value of TRUE, the next policy is evaluated. If the first policy returns the value of FALSE, the evaluation of the expression ends and subsequent policies in the expression are not evaluated.

Table 9: Policy Expression Logical Operators *(continued)*

Logical Operator	Policy Expression Logic	How Logical Operator Affects Policy Expression Evaluation
|| (Logical OR)	Logical OR requires that at least one value must be TRUE to produce output of TRUE. Routing policy value of TRUE and FALSE produces output of TRUE. Value of TRUE and TRUE produces output of TRUE. Value of FALSE and FALSE produces output of FALSE.	If the first routing policy returns the value of TRUE, the evaluation of the expression ends and subsequent policies in the expression are not evaluated. If the first policy returns the value of FALSE, the next policy is evaluated.
! (Logical NOT)	Logical NOT reverses value of TRUE to FALSE and of FALSE to TRUE. It also reverses the actions of accept and next policy to reject, and reject to accept.	If used with the logical AND operator and the first routing policy value of FALSE is reversed to TRUE, the next policy is evaluated. If the value of TRUE is reversed to FALSE, the evaluation of the expression ends and subsequent policies in the expression are not evaluated. If used with the logical OR operator and the first routing policy value of FALSE is reversed to TRUE, the evaluation of the expression ends and subsequent policies in the expression are not evaluated. If the value of TRUE is reversed to FALSE, the next policy is evaluated. If used with a policy and the flow control action is accept or next policy, these actions are reversed to reject. If the flow control action is reject, this action is reversed to accept.

For more information, see the following sections:

- Policy Expression Examples on page 66
- How a Policy Expression Is Evaluated on page 67
- Example: Evaluating Policy Expressions on page 68

Policy Expression Examples

The following examples show how to use the logical operators to create policy expressions:

- Logical AND—In the following example, **policy1** is evaluated first. If after **policy1** is evaluated, a value of TRUE is returned, **policy2** is evaluated. If a value of FALSE is returned, **policy2** is not evaluated.

  ```
  export (policy1 && policy2)
  ```

- Logical OR—In the following example, **policy1** is evaluated first. If after **policy1** is evaluated, a value of TRUE is returned, **policy2** is not evaluated. If a value of FALSE is returned, **policy2** is evaluated.

  ```
  export (policy1 || policy2)
  ```

- Logical OR and logical AND—In the following example, **policy1** is evaluated first. If after **policy1** is evaluated, a value of TRUE is returned, **policy2** is skipped and **policy3** is evaluated. If after **policy1** is evaluated, a value of FALSE is returned, **policy2** is evaluated.

If **policy2** returns a value of TRUE, **policy3** is evaluated. If **policy2** returns a value of FALSE, **policy3** is not evaluated.

```
export [(policy1 || policy2) && policy3]
```

- Logical NOT—In the following example, **policy1** is evaluated first. If after **policy1** is evaluated, a value of TRUE is returned, the value is reversed to FALSE and **policy2** is not evaluated. If a value of FALSE is returned, the value is reversed to TRUE and **policy2** is evaluated.

```
export (!policy1 && policy2)
```

The sequential list **[policy1 policy2 policy3]** is not the same as the policy expression **(policy1 && policy2 && policy3)**.

The sequential list is evaluated on the basis of a route matching a routing policy. For example, if **policy1** matches and the action is **accept** or **reject**, **policy2** and **policy3** are not evaluated. If **policy1** does not match, **policy2** is evaluated and so on until a match occurs and the action is **accept** or **reject**.

The policy expressions are evaluated on the basis of the action in a routing policy that is converted to the value of TRUE or FALSE and the logic of the specified logical operator. (For complete information about policy expression evaluation, see "How a Policy Expression Is Evaluated" on page 67.) For example, if **policy1** returns a value of FALSE, **policy2** and **policy3** are not evaluated. If **policy1** returns a value of TRUE, **policy2** is evaluated. If **policy2** returns a value of FALSE, **policy3** is not evaluated. If **policy2** returns a value of TRUE, **policy3** is evaluated.

You can also combine policy expressions and sequential lists. In the following example, if **policy1** returns a value of FALSE, **policy2** is evaluated. If **policy2** returns a value of TRUE and contains a **next policy** action, **policy3** is evaluated. If **policy2** returns a value of TRUE but does not contain an action, including a **next policy** action, **policy3** is still evaluated (because if you do not specify an action, next term or next policy are the default actions). If **policy2** returns a value of TRUE and contains an **accept** action, **policy3** is not evaluated.

```
export [(policy1 || policy2) policy3]
```

How a Policy Expression Is Evaluated

During evaluation, the policy framework software converts policy actions to values of TRUE or FALSE, which are factors in determining the flow control action that is performed upon a route. However, the software does not actually perform a flow control action on a route until it evaluates an entire policy expression.

The policy framework software evaluates a policy expression as follows:

1. The software evaluates a route against the first routing policy in a policy expression and converts the specified or default action to a value of TRUE or FALSE. (For information about the policy action conversion values, see Table 8 on page 65.)
2. The software takes the value of TRUE or FALSE and evaluates it against the logical operator used in the policy expression (see Table 9 on page 65). Based upon the logical operator used, the software determines whether or not to evaluate the next policy, if one is present.

The policy framework software uses a shortcut method of evaluation: if the result of evaluating a policy predetermines the value of the entire policy expression, the software does not evaluate the subsequent policies in the expression. For example, if the policy expression uses the logical AND operator and the evaluation of a policy returns the value of FALSE, the software does not evaluate subsequent policies in the expression because the final value of the expression is guaranteed to be FALSE no matter what the values of the unevaluated policies.

3. The software performs Step 1 and Step 2 for each subsequent routing policy in the policy expression, if they are present and it is necessary to evaluate them.
4. After evaluating the last routing policy, if it is appropriate, the software evaluates the value of TRUE or FALSE obtained from each routing policy evaluation. Based upon the logical operator used, it calculates an output of TRUE or FALSE.
5. The software converts the output of TRUE or FALSE back to an action. (For information about the policy action conversion values, see Table 8 on page 65.) The action is performed.

   If each policy in the expression returned a value of TRUE, the software converts the output of TRUE back to the flow control action specified in the last policy. For example, if the policy expression **(policy1 && policy2)** is specified and **policy1** specifies **accept** and **policy2** specifies **next term**, the **next term** action is performed.

   If an action specified in one of the policies manipulates a route characteristic, the policy framework software carries the new route characteristic forward during the evaluation of the remaining policies. For example, if the action specified in the first policy of a policy expression sets a route's metric to 500, this route matches the criteria of **metric 500** defined in the next policy. However, if a route characteristic manipulation action is specified in a policy located in the middle or the end of a policy expression, it is possible, because of the shortcut evaluation, that the policy is never evaluated and the manipulation of the route characteristic never occurs.

Example: Evaluating Policy Expressions

The following sample routing policy uses three policy expressions:

```
[edit]
policy-options {
   policy-statement policy-A {
      from {
         route-filter 10.10.0.0/16 orlonger;
      }
      then reject;
   }
}
policy-options {
   policy-statement policy-B {
      from {
         route-filter 10.20.0.0/16 orlonger;
      }
      then accept;
   }
}
```

```
protocols {
  bgp {
    neighbor 192.168.1.1 {
      export (policy-A && policy-B);
    }
    neighbor 192.168.2.1 {
      export (policy-A || policy-B);
    }
    neighbor 192.168.3.1 {
      export (!policy-A);
    }
  }
}
```

The policy framework software evaluates the transit BGP route **10.10.1.0/24** against the three policy expressions specified in the sample routing policy as follows:

- **(policy-A && policy-B)**—**10.10.1.0/24** is evaluated against **policy-A**. **10.10.1.0/24** matches the route list specified in **policy-A**, so the specified action of **reject** is returned. **reject** is converted to a value of FALSE, and FALSE is evaluated against the specified logical AND. Because the result of FALSE is certain no matter what the results of the evaluation of **policy-B** are (in policy expression logic, any result AND a value of FALSE produces the output of FALSE), **policy-B** is not evaluated and the output of FALSE is produced. The FALSE output is converted to **reject**, and **10.10.1.0/24** is rejected.
- **(policy-A || policy-B)**—**10.10.1.0/24** is evaluated against **policy-A**. **10.10.1.0/24** matches the route list specified in **policy-A**, so the specified action of **reject** is returned. **reject** is converted to a value of FALSE, then FALSE is evaluated against the specified logical OR. Because logical OR requires at least one value of TRUE to produce an output of TRUE, **10.10.1.0/24** is evaluated against **policy-B**. **10.10.1.0/24** does not match **policy-B**, so the default action of **next-policy** is returned. The **next-policy** is converted to a value of TRUE, then the value of FALSE (for **policy-A** evaluation) and TRUE (for **policy-B** evaluation) are evaluated against the specified logical OR. In policy expression logic, FALSE OR TRUE produce an output of TRUE. The output of TRUE is converted to **next-policy**. (TRUE is converted to **next-policy** because **next-policy** was the last action retained by the policy framework software.) **policy-B** is the last routing policy in the policy expression, so the action specified by the default export policy for BGP is taken.
- **(!policy-A)**—**10.10.1.0/24** is evaluated against **policy-A**. **10.10.1.0/24** matches the route list specified in **policy-A**, so the specified action of **reject** is returned. **reject** is converted to a value of FALSE, and FALSE is evaluated against the specified logical NOT. The value of FALSE is reversed to an output of TRUE based on the rules of logical NOT. The output of TRUE is converted to **accept**, and route **10.10.1.0/24** is accepted.

Applying Routing Policies to the Forwarding Table

To apply an export routing policy to the forwarding table, include the **export** statement:

```
export [ policy-names ];
```

You can include this statement at the following hierarchy levels:

- **[edit routing-options forwarding-table]**

- **[edit logical-systems *logical-system-name* routing-options forwarding-table]**

In the **export** statement, list the name of the routing policy to be evaluated when routes are being exported from the routing table into the forwarding table. Only active routes are exported from the routing table.

You can reference the same routing policy one or more times in the same or a different **export** statement.

For information about how the policy framework software evaluates a routing policy, see "How a Routing Policy Is Evaluated" on page 22.

You can apply export policies to routes being exported from the routing table into the forwarding table for the following features:

- Per-packet load balancing
- Class of service (CoS)

For more information about per-packet load balancing, see "Configuring Per-Packet Load Balancing" on page 160.

Related Documentation

- *Junos OS Class of Service Configuration Guide*

Configuring Dynamic Routing Policies

The verification process required to commit configuration changes can entail a significant amount of overhead and time. For example, changing a prefix in one line of a routing policy that is 20,000 lines long can take up to 20 seconds to commit. It can be useful to be able to commit routing policy changes much more quickly.

In Junos OS Release 9.5 and later, you can configure routing policies and certain routing policy objects in a dynamic database that is not subject to the same verification required in the standard configuration database. As a result, the time it takes to commit changes to the dynamic database is much shorter than for the standard configuration database. You can then reference these policies and policy objects in routing policies you configure in the standard database. BGP is the only protocol to which you can apply routing policies that reference policies and policy objects configured in the dynamic database. After you configure and commit a routing policy based on the objects configured in the dynamic database, you can quickly update any existing routing policy by making changes to the dynamic database configuration.

CAUTION: **Because the Junos OS does not validate configuration changes to the dynamic database, when you use this feature, you should test and verify all configuration changes before committing them.**

This section discusses the following topics:

Configuring Routing Policies and Policy Objects in the Dynamic Database

Junos OS Release 9.5 and later support a configuration database, the *dynamic database*, which can be edited in a similar way to the standard configuration database but which is not subject to the same verification process to commit configuration changes. As a result, the time it takes to commit a configuration change is much faster. The policies and policy objects defined in the dynamic database can then be referenced in routing policies configured in the standard configuration. The dynamic database is stored in the **/var/run/db/juniper.dyn** directory.

To configure the dynamic database, enter the **configure dynamic** command to enter the configuration mode for the dynamic database:

```
user@host> configure dynamic
Entering configuration mode

[edit dynamic]
user@host#
```

In this dynamic configuration database, you can configure the following statements at the **[edit policy-options]** hierarchy level:

- **as-path** *name*
- **as-path-group** *group-name*
- **community** *community-name*
- **condition** *condition-name*
- **prefix-list** *prefix-list-name*
- **policy-statement** *policy-statement-name*

NOTE: No other configuration is supported at the **[edit dynamic]** hierarchy level.

Use the **policy-statement** *policy-statement-name* statement to configure routing policies as you would in the standard configuration database.

To exit configuration mode for the dynamic database, issue the **exit configuration-mode** command from any level within the **[edit dynamic]** hierarchy, or use the **exit** command from the top level.

Configuring Routing Policies Based on Dynamic Database Configuration

In the standard configuration mode, you can configure routing policies that reference policies and policy objects configured at the **[edit dynamic]** hierarchy level in the dynamic database. To define a routing policy that references the dynamic database configuration, include the **dynamic-db** statement at the **[edit policy-options policy-statement *policy-statement-name*]** hierarchy level:

```
[edit policy-options]
policy-statement policy-statement-name {
  dynamic-db;
}
```

You can also define specific policy objects based on the configuration of these objects in the dynamic database. To define a policy object based on the dynamic database, include the **dynamic-db** statement with the following statements at the **[edit policy-options]** hierarchy level:

- **as-path** *name*
- **as-path-group** *group-name*
- **community** *community-name*
- **condition** *condition-name*
- **prefix-list** *prefix-list-name*

In the standard configuration, you can also define a routing policy that references any policy object you have configured in the standard configuration that references an object configured in the dynamic database.

For example, in standard configuration mode, you configure a prefix list **prefix-list pl2** that references a prefix list, also named **prefix-list pl2**, that has been configured in the dynamic database:

```
[edit policy-options]
prefix-list pl2 {
  dynamic-db; # Reference a prefix list configured in the dynamic database.
}
```

You then configure a routing policy in the standard configuration that includes **prefix-list pl2**:

```
[edit policy-options]
policy-statement one {
  term term1 {
    from {
      prefix-list pl2; # Include the prefix list configured in the standard configuration
        # database, but which references a prefix list configured in the dynamic database.
    }
    then accept;
```

```
    }
    then reject;
}
```

If you need to update the configuration of **prefix-list pl2**, you do so in the dynamic database configuration using the **[edit dynamic]** hierarchy level. This enables you to make commit configuration changes to the prefix list more quickly than you can in the standard configuration database.

NOTE: If you are downgrading the Junos OS to Junos OS Release 9.4 or earlier, you must first delete any routing policies that reference the dynamic database. That is, you must delete any routing policies or policy objects configured with the **dynamic-db** statement.

Applying Dynamic Routing Policies to BGP

BGP is the only routing protocol to which you can apply routing policies that reference the dynamic database configuration. You must apply these policies in the standard configuration. Dynamic policies can be applied to BGP export or import policy. They can also be applied at the global, group, or neighbor hierarchy level.

To apply a BGP export policy, include the **export [** *policy-names* **]** statement at the **[edit protocols bgp]**, **[edit protocols bgp group** *group-name***]**, or **[edit protocols bgp group** *group-name* **neighbor** *address***]** hierarchy level.

```
[edit]
protocols
  bgp {
    export [ policy-names ];
  }
}
```

To apply a BGP import policy, include the **import [** *policy-names* **]** statement at the **[edit protocols bgp]**, **[edit protocols bgp group** *group-name***]**, or **[edit protocols bgp group** *group-name* **neighbor** *address***]** hierarchy level.

```
[edit]
protocols
  bgp {
    import [ policy-names ];
  }
}
```

Include one or more policy names configured in that standard configuration at the **[edit policy-options** *policy-statement***]** hierarchy level that reference policies configured in the dynamic database.

Preventing Reestablishment of BGP Peering Sessions After NSR Routing Engine Switchover

If you have active nonstop routing (NSR) enabled, the dynamic database is not synchronized with the backup Routing Engine. As a result, if a switchover to a backup Routing Engine occurs, import and export policies running on the master Routing Engine

at the time of the switchover might no longer be available. Therefore, you might want to prevent a BGP peering session from automatically being reestablished as soon as a switchover occurs.

You can configure the router not to reestablish a BGP peering session after an active nonstop routing switchover either for a specified period or until you manually reestablish the session. Include the **idle-after-switch-over (*seconds* | forever)** statement at the **[edit protocols bgp]**, **[edit protocols bgp group *group-name*]**, or **[edit protocols bgp group *group-name* neighbor *address*]** hierarchy level:

```
[edit]
bgp {
  protocols {
    idle-after-switch-over (seconds | never);
  }
}
```

For ***seconds***, specify a value from 1 through 4,294,967,295 ($2^{32} - 1$). The BGP peering session is not reestablished until after the specified period. If you specify the **forever** option, the BGP peering session is not established until you issue the **clear bgp neighbor** command. For more information about configuring active nonstop routing, see the *Junos OS High Availability Configuration Guide*.

Example: Configuring a BGP Export Policy That References a Dynamic Routing Policy

In this example, you configure the following in the dynamic database: a routing policy, **policy-statement one**, and two prefix lists: **prefix-list pl1**, which is referenced in **policy-statement one**, and **prefix-list pl2**.

In the standard configuration database, you configure the following routing policies and policy objects: a routing policy (**policy-statement one**) that references the routing policy with the same name configured in the dynamic database; a prefix list (**prefix-list pl2**) that references the prefix list with the same name configured in the dynamic database; and a routing policy (**policy-statement two**) that references the prefix list **prefix-list pl2** you configured in the standard configuration and references the prefix list with the same name configured in the dynamic database.

You then create and apply a policy expression that includes **policy-statement one** and **policy-statement two** to BGP export policy in the standard configuration.

In the dynamic database, configure **policy-statement one**, **prefix-list pl1**, and **prefix-list pl2**:

```
[edit dynamic]
policy-options {
  prefix-list pl1 {
    8.8.0.0/16;
    12.12.12.3/32
  }
  prefix-list pl2;
    10.10.0.0/16
  }
  policy-statement one {
    term term1
      from {
```

```
            prefix-list pl1
          }
          then accept;
        }
        then reject;
      }
    }
```

In the standard configuration database, configure **policy-statement one** based on the policy with the same name configured in the dynamic database. In addition, configure **prefix-list pl2**, which references the prefix list with the same name in the dynamic database, and configure **policy-statement two**, which references **prefix-list pl2**.

```
[edit]
policy-options {
  prefix-list pl2;
    dynamic-db; # Reference 'prefix-list pl2' configured in the dynamic database.
  }
  policy-statement two {
    term term1 {
      from {
        prefix-list pl2; # Configure a routing policy that includes a prefix list that
          # references the dynamic database.
      }
      then accept;
    }
    then reject;
  }
  policy-statement one;
    dynamic-db; # Configure a policy in the standard configuration that references
      # the policy configured in the dynamic database.
  }
}
```

Apply a policy expression that includes routing policies **one** and **two** to BGP export policy for an internal BGP group **test**.

```
[edit]
protocols {
  bgp {
    group test;
      type internal;
      local-address 10.255.245.44;
      export ( one && two ); # If routing policy one returns the value of TRUE, policy # two
        is evaluated.
        # If policy one returns the value of FALSE, the evaluation of the expression # ends
        and policy two is not evaluated.
    }
  }
}
```

Forwarding Packets to the Discard Interface

The discard interface allows you to protect a network from denial-of-service (DoS) attacks by identifying the target IP address that is being attacked and configuring a policy

to forward all packets to a discard interface. All packets forwarded to the discard interface are dropped.

To configure the discard interface, include the **dsc** statement:

```
dsc {
  unit 0 {
    family inet {
      filter {
        input filter-name;
        output filter-name;
      }
    }
  }
}
```

You can include this statement at the following hierarchy levels:

- **[edit interfaces *interface-name*]**
- **[edit logical-systems *logical-system-name* interfaces *interface-name*]**

The **dsc** interface name denotes the discard interface. The discard interface supports only unit 0.

The following two configurations are required to configure a policy to forward all packets to the discard interface.

Configure an input policy to associate a community with the discard interface:

```
[edit]
policy-options {
  community community-name members [ community-id ];
  policy-statement statement-name {
    term term-name {
      from community community-name;
      then {
        next-hop address; # Remote end of the point-to-point interface
        accept;
      }
    }
  }
}
```

Configure an output policy to set up the community on the routes injected into the network:

```
[edit]
policy-options {
  policy-statement statement-name {
    term term-name {
      from prefix-list name;
      then community (set | add | delete) community-name;
    }
  }
}
```

Testing Routing Policies

Before applying a routing policy, you can issue the **test policy** command to ensure that the policy produces the results that you expect:

```
user@host> test policy policy-name prefix
```

Example: Testing a Routing Policy

Test the following policy, which looks for unwanted routes and rejects them:

```
[edit policy-options]
policy-statement reject-unwanted-routes {
   term drop-these-routes {
      from {
         route-filter 0/0 exact;
         route-filter 10/8 orlonger;
         route-filter 172.16/12 orlonger;
         route-filter 192.168/16 orlonger;
         route-filter 224/3 orlonger;
      }
      then reject;
   }
}
```

Test this policy against all routes in the routing table:

```
user@host> test policy reject-unwanted-routes 0/0
```

Test this policy against a specific set of routes:

```
user@host> test policy reject-unwanted-routes 10.49.0.0/16
```

Routing Policy Examples

The following examples show how to configure routing policies for various purposes:

Examples: Configuring Static Routes

Understanding Basic Static Routing

Routes that are permanent fixtures in the routing and forwarding tables are often configured as static routes. These routes generally do not change, and often include only one or very few paths to the destination.

To create a static route in the routing table, you must, at minimum, define the route as static and associate a next-hop address with it. The static route in the routing table is inserted into the forwarding table when the next-hop address is reachable. All traffic destined for the static route is transmitted to the next-hop address for transit.

You can specify options that define additional information about static routes that is included with the route when it is installed in the routing table. All static options are optional.

Example: Configuring a Basic Set of Static Routes

This example shows how to configure a basic set of static routes.

Requirements

In this example, no special configuration beyond device initialization is required.

Overview

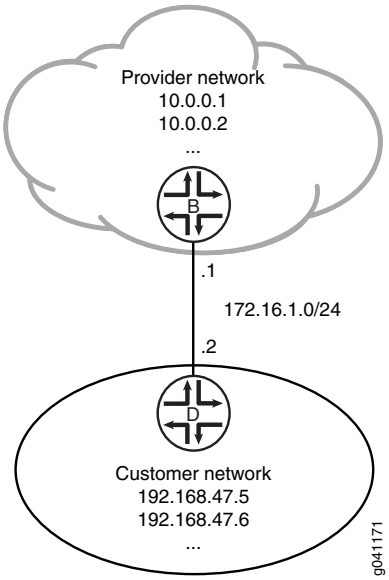
There are many practical applications for static routes. Static routing is often used at the network edge to support attachment to stub networks, which, given their single point of entry and egress, are well suited to the simplicity of a static route. In Junos OS, static routes have a global preference of 5. Static routes are activated if the specified next hop is reachable.

In this example, you configure the static route 192.168.47.0/24 from the provider network to the customer network, using the next-hop address of 172.16.1.2. You also configure a static default route of 0.0.0.0/0 from the customer network to the provider network, using a next-hop address of 172.16.1.1.

For demonstration purposes, some loopback interfaces are configured on Device B and Device D. These loopback interfaces provide addresses to ping and thus verify that the static routes are working.

Figure 11 on page 79 shows the sample network.

Figure 11: Customer Routes Connected to a Service Provider



Configuration

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the **[edit]** hierarchy level.

Device B

```
set interfaces ge-1/2/0 unit 0 description B->D
set interfaces ge-1/2/0 unit 0 family inet address 172.16.1.1/24
set interfaces lo0 unit 57 family inet address 10.0.0.1/32
set interfaces lo0 unit 57 family inet address 10.0.0.2/32
set routing-options static route 192.168.47.0/24 next-hop 172.16.1.2
```

Device D

```
set interfaces ge-1/2/0 unit 1 description D->B
set interfaces ge-1/2/0 unit 1 family inet address 172.16.1.2/24
set interfaces lo0 unit 2 family inet address 192.168.47.5/32
set interfaces lo0 unit 2 family inet address 192.168.47.6/32
set routing-options static route 0.0.0.0/0 next-hop 172.16.1.1
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see Using the CLI Editor in Configuration Mode in the *Junos OS CLI User Guide*.

To configure basic static routes:

1. On Device B, configure the interfaces.

   ```
   [edit interfaces]
   user@B# set ge-1/2/0 unit 0 description B->D
   user@B# set ge-1/2/0 unit 0 family inet address 172.16.1.1/24
   user@B# set lo0 unit 57 family inet address 10.0.0.1/32
   user@B# set lo0 unit 57 family inet address 10.0.0.2/32
   ```

2. On Device B, create a static route and set the next-hop address.

   ```
   [edit routing-options]
   user@B# set static route 192.168.47.0/24 next-hop 172.16.1.2
   ```

3. If you are done configuring Device B, commit the configuration.

   ```
   [edit interfaces]
   user@B# commit
   ```

4. On Device D, configure the interfaces.

   ```
   [edit]
   user@D# set ge-1/2/0 unit 1 description D->B
   user@D# set ge-1/2/0 unit 1 family inet address 172.16.1.2/24
   user@D# set lo0 unit 2 family inet address 192.168.47.5/32
   user@D# set lo0 unit 2 family inet address 192.168.47.6/32
   ```

5. On Device D, create a static route and set the next-hop address.

   ```
   [edit routing-options]
   user@D# set static route 0.0.0.0/0 next-hop 172.16.1.1
   ```

6. If you are done configuring Device D, commit the configuration.

   ```
   [edit]
   user@D# commit
   ```

Results

Confirm your configuration by issuing the **show interfaces** and **show routing-options** commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

Device B

```
user@B# show interfaces
ge-1/2/0 {
   unit 0 {
      description B->D;
      family inet {
         address 172.16.1.1/24;
      }
```

```
    }
  }
  lo0 {
    unit 57 {
      family inet {
        address 10.0.0.1/32;
        address 10.0.0.2/32;
      }
    }
  }

  user@B# show routing-options
  static {
    route 192.168.47.0/24 next-hop 172.16.1.2;
  }
```

Device D

```
user@D# show interfaces
ge-1/2/0 {
  unit 1 {
    description D->B;
    family inet {
      address 172.16.1.2/24;
    }
  }
}
lo0 {
  unit 2 {
    family inet {
      address 192.168.47.5/32;
      address 192.168.47.6/32;
    }
  }
}

user@D# show routing-options
static {
  route 0.0.0.0/0 next-hop 172.16.1.1;
}
```

Verification

Confirm that the configuration is working properly.

- Checking the Routing Tables on page 81
- Pinging the Remote Addresses on page 82

Checking the Routing Tables

Purpose Make sure that the static routes appear in the routing tables of Device B and Device D.

Action

```
user@B> show route
inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.0.1/32        *[Direct/0] 00:29:43
                    > via lo0.57
```

```
10.0.0.2/32        *[Direct/0] 00:29:43
                    > via lo0.57
172.16.1.0/24      *[Direct/0] 00:34:40
                    > via ge-1/2/0.0
172.16.1.1/32      *[Local/0] 00:34:40
                      Local via ge-1/2/0.0
192.168.47.0/24   *[Static/5] 00:31:23
        > to 172.16.1.2 via ge-1/2/0.0

user@D> show route
inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0       *[Static/5] 00:31:24
        > to 172.16.1.1 via ge-1/2/0.1
172.16.1.0/24      *[Direct/0] 00:35:21
                    > via ge-1/2/0.1
172.16.1.2/32      *[Local/0] 00:35:21
                      Local via ge-1/2/0.1
192.168.47.5/32    *[Direct/0] 00:35:22
                    > via lo0.2
192.168.47.6/32    *[Direct/0] 00:35:21
                    > via lo0.2
```

Meaning The static routes are in the routing tables.

Pinging the Remote Addresses

Purpose Verify that the static routes are working.

From Device B, ping one of the loopback interface addresses on Device D.

From Device D, ping one of the loopback interface addresses on Device B.

Action

```
user@B> ping 192.168.47.5
PING 192.168.47.5 (192.168.47.5): 56 data bytes
64 bytes from 192.168.47.5: icmp_seq=0 ttl=64 time=156.126 ms
64 bytes from 192.168.47.5: icmp_seq=1 ttl=64 time=120.393 ms
64 bytes from 192.168.47.5: icmp_seq=2 ttl=64 time=175.361 ms

user@D> ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1): 56 data bytes
64 bytes from 10.0.0.1: icmp_seq=0 ttl=64 time=1.315 ms
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=31.819 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=1.268 ms
```

Example: Configuring IPv6 Static Routes

This example shows how to configure static routes when the interfaces have IPv6 addresses.

Requirements

In this example, no special configuration beyond device initialization is required.

Overview

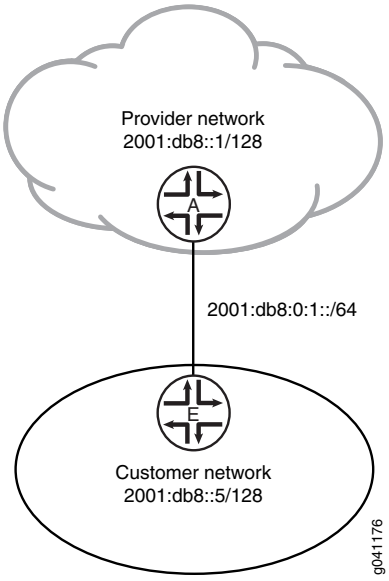
There are many practical applications for static routes. Static routing is often used at the network edge to support attachment to stub networks, which, given their single point of entry and egress, are well suited to the simplicity of a static route. In Junos OS, static routes have a global preference of 5. Static routes are activated if the specified next hop is reachable.

In this example, you configure a static default route of ::/0, using a next-hop address 2001:db8:0:1:2a0:a502:0:1da.

For demonstration purposes, some loopback interfaces are configured on Device A and Device E. These loopback interfaces provide addresses to ping and thus verify that the static routes are working.

Figure 12 on page 83 shows the sample network.

Figure 12: Customer Routes Connected to a Service Provider



Configuration

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the **[edit]** hierarchy level.

Device A

```
set interfaces fe-1/2/0 unit 1 description to-E
set interfaces fe-1/2/0 unit 1 family inet6 address 2001:db8:0:1:2a0:a502:0:1da/64
set interfaces lo0 unit 1 family inet6 address 2001:db8::1/128 primary
set interfaces lo0 unit 1 family inet6 address 2001:db8::2/128
```

```
set interfaces lo0 unit 1 family inet6 address 2001:db8::3/128
set routing-options rib inet6.0 static route 2001:db8::5/128 next-hop
  2001:db8:0:1:2a0:a502:0:19da
```

Device E

```
set interfaces fe-1/2/0 unit 25 description to-A
set interfaces fe-1/2/0 unit 25 family inet6 address 2001:db8:0:1:2a0:a502:0:19da/64
set interfaces lo0 unit 5 family inet6 address 2001:db8::5/128
set routing-options rib inet6.0 static route ::/0 next-hop 2001:db8:0:1:2a0:a502:0:1da
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see Using the CLI Editor in Configuration Mode in the *Junos OS CLI User Guide*.

To configure basic static routes:

1. On Device A, configure the interfaces.

```
[edit interfaces]
set interfaces fe-1/2/0 unit 1 description to-E
set interfaces fe-1/2/0 unit 1 family inet6 address 2001:db8:0:1:2a0:a502:0:1da/64

set interfaces lo0 unit 1 family inet6 address 2001:db8::1/128 primary
set interfaces lo0 unit 1 family inet6 address 2001:db8::2/128
set interfaces lo0 unit 1 family inet6 address 2001:db8::3/128
```

2. On Device A, create a static route to Device E's loopback address and set the next-hop address.

   This ensures that Device A has a route back to Device E.

```
[edit routing-options]
set rib inet6.0 static route 2001:db8::5/128 next-hop 2001:db8:0:1:2a0:a502:0:19da
```

3. If you are done configuring Device A, commit the configuration.

```
[edit interfaces]
user@A# commit
```

4. On Device E, configure the interfaces.

```
[edit]
set interfaces fe-1/2/0 unit 25 description to-A
set interfaces fe-1/2/0 unit 25 family inet6 address 2001:db8:0:1:2a0:a502:0:19da/64
set interfaces lo0 unit 5 family inet6 address 2001:db8::5/128
```

5. On Device E, create a static default route and set the next-hop address.

```
[edit routing-options]
set routing-options rib inet6.0 static route ::/0 next-hop 2001:db8:0:1:2a0:a502:0:1da
```

6. If you are done configuring Device E, commit the configuration.

```
[edit]
user@E# commit
```

Results

Confirm your configuration by issuing the **show interfaces** and **show routing-options** commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

Device A

```
user@A# show interfaces
fe-1/2/0 {
  unit 1 {
    description to-E;
    family inet6 {
      address 2001:db8:0:1:2a0:a502:0:1da/64;
    }
  }
}
lo0 {
  unit 1 {
    family inet6 {
      address 2001:db8::1/128 {
        primary;
      }
      address 2001:db8::2/128;
      address 2001:db8::3/128;
    }
  }
}

user@A# show routing-options
rib inet6.0 {
  static {
    route 2001:db8::5/128 next-hop 2001:db8:0:1:2a0:a502:0:19da;
  }
}
```

Device E

```
user@E# show interfaces
fe-1/2/0 {
  unit 25 {
    description to-A;
    family inet6 {
      address 2001:db8:0:1:2a0:a502:0:19da/64;
    }
  }
}
lo0 {
  unit 5 {
    family inet6 {
      address 2001:db8::5/128;
    }
  }
}

user@E# show routing-options
rib inet6.0 {
  static {
    route ::/0 next-hop 2001:db8:0:1:2a0:a502:0:1da;
  }
}
```

Verification

Confirm that the configuration is working properly.

- Checking the Routing Tables on page 86
- Pinging the Remote Addresses on page 86

Checking the Routing Tables

Purpose Make sure that the static routes appear in the routing tables of Device A and Device E.

Action

```
user@A> show route protocol static
inet6.0: 9 destinations, 9 routes (9 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

2001:db8::5/128    *[Static/5] 00:27:46
                    > to 2001:db8:0:1:2a0:a502:0:19da via fe-1/2/0.1

user@E> show route protocol static
inet6.0: 7 destinations, 7 routes (7 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

::/0               *[Static/5] 00:19:11
                    > to 2001:db8:0:1:2a0:a502:0:1da via fe-1/2/0.25
```

Meaning The static routes are in the routing tables.

Pinging the Remote Addresses

Purpose Verify that the static routes are working.

From Device A, ping one of the loopback interface addresses on Device E.

From Device E, ping one of the loopback interface addresses on Device A.

Action

```
user@A> ping 2001:db8::5
PING6(56=40+8+8 bytes) 2001:db8:0:1:2a0:a502:0:1da --> 2001:db8::5
16 bytes from 2001:db8::5, icmp_seq=0 hlim=64 time=1.790 ms
16 bytes from 2001:db8::5, icmp_seq=1 hlim=64 time=1.529 ms
16 bytes from 2001:db8::5, icmp_seq=2 hlim=64 time=1.531 ms

user@E> ping 2001:db8::3
PING6(56=40+8+8 bytes) 2001:db8:0:1:2a0:a502:0:19da --> 2001:db8::3
16 bytes from 2001:db8::3, icmp_seq=0 hlim=64 time=2.146 ms
16 bytes from 2001:db8::3, icmp_seq=1 hlim=64 time=1.964 ms
16 bytes from 2001:db8::3, icmp_seq=2 hlim=64 time=1.550 ms
```

Example: Defining a Routing Policy from BGP to IS-IS

Accept BGP routes advertised by the peer **192.168.1.1**. If a route matches, it is accepted, and no further evaluation is performed on that route. If a route does not match, the accept or reject action specified by the default policy is taken. (For more information about the default routing policies, see "Default Routing Policies and Actions" on page 16.) If you

apply this routing policy to imported BGP routes, only the routes learned from the peer **192.168.1.1** and BGP transit routes are accepted from BGP peers.

```
[edit]
policy-options {
   policy-statement bgp-to-isis {
      term term1 {
         from {
            neighbor 192.168.1.1;
         }
         then {
            accept;
         }
      }
   }
}
```

Example: Using Routing Policy to Set a Preference

Define a routing policy which matches routes from specific next hops that are being advertised to specific neighbors and which sets a preference. If a route does not match the first term, it is evaluated by the second term. If it still does not match, the next routing policy, if configured, is evaluated; then the accept or reject action specified by the default policy is taken. (For more information about the default routing policies, see "Default Routing Policies and Actions" on page 16.)

```
[edit]
policy-options {
   policy-statement set-preference {
      term term1 {
         from {
            next-hop [ 10.0.0.1 10.0.0.2 ];
         }
         to {
            neighbor 192.168.1.1;
         }
         then {
            preference 10;
         }
      }
      term term2 {
         from {
            next-hop 10.0.0.3;
         }
         to {
            neighbor 192.168.1.1;
         }
         then {
            preference 15;
         }
      }
   }
}
```

Example: Importing and Exporting Access and Access-Internal Routes in a Routing Policy

Configure import and export of access routes and access-internal routes in a routing policy. These routes are used by the DHCP application on a video services router to represent either the end users or the networks behind the attached video services router. (For more information about configuring DHCP relay on the router, see *Junos OS Subscriber Access Configuration Guide*.)

An access route represents a network behind an attached video services router, and is set to a preference of 13. An access-internal route is a /32 route that represents a directly attached end user, and is set to a preference of 12.

```
[edit]
policy-options {
   policy-statement foo {
      term term1 {
         from protocol {
            access;
            access-internal;
         }
         then accept;
      }
   }
}
```

Example: Exporting Routes to IS-IS

Configure the router to export to IS-IS the routes that match the **dmz** and **local-customers** routing policies.

```
[edit]
protocols {
   isis {
      export [ dmz local-customers ];
   }
}
```

Example: Applying Export and Import Policies to BGP Peer Groups

For three BGP peer groups, apply various export and import filters.

```
[edit]
protocols {
   bgp {
      group 1 {
         type external;
         peer-as 47;
         export local-customers;
         import [ martian-filter long-prefix-filter as47-filter ];
         neighbor 192.168.1.4;
         neighbor 192.168.1.5;
      }
```

```
    group 2 {
      type external;
      peer-as 42;
      export local-customers;
      import [ martian-filter long-prefix-filter as42-filter ];
      neighbor 192.168.1.4;
      neighbor 192.168.1.5;
    }
    group 3 {
      type internal;
      export local-customers;
      neighbor 10.1.1.1;
    }
  }
}
```

Example: Defining a Routing Policy Based on the Number of BGP Communities

Create a policy that accepts BGP routes based on the number of BGP communities. In this example, routes can contain two, three, or four communities to be considered a match. For example, if a route contains three communities, it is considered a match and is accepted. If a route contains one community, it is not considered a match and is rejected.

NOTE: The **community-count** attribute only works with standard communities. It does not work with extended communities.

```
[edit]
policy-options {
  policy-statement import-bgp {
    term community {
      from {
        community-count 2 orhigher;
        community-count 4 orlower;
      }
      then {
        accept;
      }
    }
  }
}
```

Example: Applying a Prefix to Routes Learned from a Peer

Apply the **long-prefix-filter** prefix only to routes learned from a particular peer within a group.

```
[edit]
protocols {
  bgp {
    group 4 {
      type external;
      peer-as 47;
```

```
        export local-customers;
        import [ martian-filter as47-filter ];
        neighbor 192.168.1.4;
        neighbor 192.168.1.5;
        neighbor 192.168.1.6 {
          import [ martian-filter as47-filter long-prefix-filter ];
        }
      }
    }
}
```

Example: Redistributing BGP Routes with a Specific Community Tag into IS-IS

Redistribute BGP routes with a community tag of **444:5** into IS-IS, changing the metric to **14**.

```
[edit]
protocols {
  isis {
    export edu-to-isis;
  }
}
policy-options {
  community edu members 444:5;
  policy-statement edu-to-isis {
    from {
      protocol bgp;
      community edu;
    }
    then {
      metric 14;
      accept;
    }
  }
}
```

Example: Redistributing OSPF Routes into BGP

Redistribute OSPF routes from Area 1 only into BGP, and do not advertise routes learned by BGP.

```
[edit]
routing-options {
  autonomous-system 56;
}
protocols {
  bgp {
    export ospf-into-bgp;
    group {
      type external;
      peer-as 23;
      allow {
        0.0.0.0/0;
      }
    }
```

```
    }
  }
  policy-options {
    policy-statement ospf-into-bgp {
      term ospf-only {
        from {
          protocol ospf;
          area 1;
          then accept;
        }
      }
    }
  }
```

Example: Exporting Direct Routes Into IS-IS

Export direct routes into IS-IS for all interfaces, even if IS-IS is not configured on an interface.

```
[edit]
protocols {
  isis {
    export direct-routes;
  }
}
policy-options {
  policy-statement direct-routes {
    from protocol direct;
    then accept;
  }
}
```

Example: Exporting Internal IS-IS Level 1 Routes to Level 2

Export IS-IS Level 1 internal-only routes into Level 2.

```
[edit]
protocols {
  isis {
    export L1-L2;
  }
}
policy-statement L1-L2 {
  term one {
    from {
      level 1;
      external;
    }
    then reject;
  }
  term two {
    from level 1;
    to level 2;
    then accept;
  }
```

```
}
```

Example: Exporting IS-IS Level 2 Routes to Level 1

Export IS-IS Level 2 routes into Level 1.

```
[edit]
protocols {
   isis {
      export L2-L1;
   }
}
policy-statement L2-L1 {
   term one {
      from level 2;
      to level 1;
      then accept;
   }
}
```

Example: Assigning Different Forwarding Next-Hop LSPs to Different Destination Prefixes

Assign different forwarding next-hop LSPs to different destination prefixes learned from BGP.

```
routing-options {
   router-id 10.10.20.101;
   autonomous-system 2;
   forwarding-table {
      export forwarding-policy;
   }
}
policy-options {
   policy-statement forwarding-policy {
      term one {
         from {
            protocol bgp;
            route-filter 10.1.0.0/16 orlonger;
         }
         then {
            install-nexthop lsp mc-c-lsp-1;
            accept;
         }
      }
      term two {
         from {
            protocol bgp;
            route-filter 10.2.0.0/16 orlonger;
         }
         then {
            install-nexthop lsp mc-c-lsp-2;
            accept;
         }
      }
```

```
    term three {
      from {
        protocol bgp;
        route-filter 10.3.0.0/16 orlonger;
      }
      then {
        install-nexthop lsp mc-c-lsp-3;
        accept;
      }
    }
  }
}
protocols {
  mpls {
    label-switched-path mc-c-lsp-1 {
      from 10.10.20.101;
      to 10.10.20.103;
    }
    label-switched-path mc-c-lsp-2 {
      from 10.10.20.101;
      to 10.10.20.103;
    }
    label-switched-path mc-c-lsp-3 {
      from 10.10.20.101;
      to 10.10.20.103;
    }
  }
}
```

Example: Grouping Destination Prefixes

Configure a routing policy to group destination prefixes.

```
[edit]
policy-options {
  policy-statement set-dest-class {
    term 1 {
      from community nets1;
      then {
        destination-class on-net;
        accept;
      }
    }
    term 2 {
      from community nets2;
      then {
        destination-class off-net;
        accept;
      }
    }
  }
  community nets1 [7:8 9:10];
  community nets2 [1:2 4:5];
}
```

Apply a routing policy to the forwarding table with the corresponding destination class.

```
[edit]
routing-options {
  forwarding-table {
    export set-dest-class;
  }
}
```

Enable packet counting on an interface.

```
[edit interfaces]
interfaces so-1/0/1 {
  unit 0 {
    family inet6 {
      accounting {
        destination-class-usage;
      }
    }
  }
}
```

Example: Grouping Source Prefixes

Configure a routing policy to group source prefixes, and allow prefixes that match the policy statement to have a source class created for them.

```
[edit]
policy-options {
  policy-statement set-gold-class {
    term {
      from
      route-filter 10.210.0.0/16 orlonger;
      route-filter 10.215.0.0/16 orlonger;
      then {
        source-class gold-class;
      }
    }
  }
}
```

Apply a routing policy to the forwarding table with the corresponding source class.

```
[edit]
routing-options {
  forwarding-table {
    export set-gold-class;
  }
}
```

Enable packet counting on an interface. In this example, one interface accommodates both input and output.

```
[edit interfaces]
interfaces ge/0/0/0 {
  unit 0 {
    family inet {
      accounting {
        source-class-usage {
```

```
            input;
            output;
          }
        }
      }
    }
}
```

Example: Grouping Source and Destination Prefixes in a Forwarding Class

Configure a routing policy to group source and destination prefixes in a forwarding class.

```
[edit]
policy-options {
  policy-statement set-bronze-class {
    term {
      from
      route-filter 10.210.0.0/16 orlonger;
      route-filter 10.215.0.0/16 orlonger;
      then {
        forwarding-class bronze-class;
      }
    }
  }
}
```

Apply a routing policy to the forwarding table with the corresponding forwarding class.

```
[edit]
routing-options {
  forwarding-table {
    export set-bronze-class;
  }
}
```

Enable counting of incoming source packets on an interface.

```
[edit interfaces]
interfaces fe/1/0/0 {
  unit 0 {
    family inet {
      accounting {
        source-class-usage {
          input;
        }
      }
    }
  }
}
interfaces fe/1/0/1 {
  unit 0 {
    family inet {
      accounting {
        source-class-usage {
          output;
        }
```

```
        }
      }
    }
  }
  interfaces fe/1/0/2 {
    unit 0 {
      family inet {
        accounting {
          destination-class-usage;
        }
      }
    }
  }
```

Example: Accepting Routes with Specific Destination Prefixes

Accept routes with destination prefixes **201:db8::8000/32** and **201:db8::8001/32**.

```
[edit policy-options]
policy-statement export-exact {
  term a {
    from {
      route-filter 201:db8::8000/32 exact;
      route-filter 201:db8::8001/32 exact;
    }
    then {
      accept;
    }
  }
  term b {
    then {
      reject;
    }
  }
}
```

Example: Accepting Routes from BGP with a Specific Destination Prefix

Accepts routes from BGP that have destination prefix **201:db8::8000/32**.

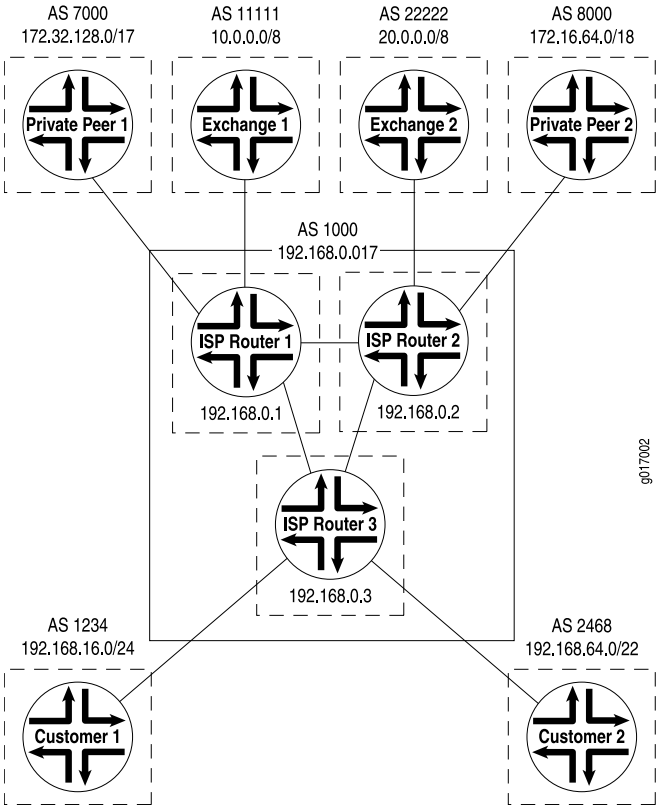
```
[edit policy-options]
policy-statement export-exact {
  term a {
    from {
      protocol bgp;
      route-filter 201:db8::8000/32 exact;
    }
    then {
      accept;
    }
  }
  term b {
    then {
      reject;
    }
```

```
    }
}
```

Example: Using Routing Policy in an ISP Network

This section provides an example of how policies might be used in a typical Internet service provider (ISP) network. In this network example (see Figure 13 on page 97), the ISP's AS number is 1000. The ISP has two transit peers (AS 11111 and AS 22222) to which it connects at an exchange point. The ISP is also connected to two private peers (AS 7000 and AS 8000) with which it exchanges specific customer routes. The ISP has two customers (AS 1234 and AS 2468) to which it connects using BGP.

Figure 13: ISP Network Example



In this example, the ISP policies are configured in an outbound direction; that is, the example focuses on the routes that the ISP announces to its peers and customers, and includes the following:

1. The ISP has been assigned AS 1000 and the routing space of **192.168.0/17**. With the exception of the two customer networks shown in Figure 13 on page 97, all other customer routes are simulated with static routes.
2. The ISP has connectivity to two different exchange peers: AS 11111 and AS 22222. These peers are used for transit service to other portions of the Internet. This means that the ISP is accepting all routes (the full Internet routing table) from those BGP peers. To

help maintain an optimized Internet routing table, the ISP is configured to advertise only two aggregate routes to the transit peers.

3. The ISP also has direct connectivity to two private peers: AS 7000 and AS 8000. The ISP administrators want all data to the private peers to use this direct link. As a result, all the customer routes from the ISP are advertised to those private peers. These peers then advertise all their customer routes to the ISP.
4. Finally, the ISP has two customers with which it communicates using BGP: AS 1234 and AS 2468. Each customer has a different set of requirements.

The following sections discuss the following topics:

Requesting a Single Default Route on the Customer 1 Router

Customer 1 has only a single route to the ISP and is using the ISP for transit service. This customer has requested a single default route (**0.0.0.0/0**) from the ISP.

```
[edit]
interfaces {
   so-0/0/1 {
      description "Connection to ISP Router 3";
      unit 0 {
         family inet {
            address 10.222.70.1/30;
         }
      }
   }
   fxp0 {
      description "MGMT INTERFACE - DO NOT DELETE";
      unit 0 {
         family inet {
            address 10.251.0.9/24;
         }
      }
   }
   lo0 {
      unit 0 {
         family inet {
            address 192.168.16.1/32;
         }
      }
   }
}
routing-options {
   static {
```

```
        route 192.168.16.0/27 reject;
        route 192.168.16.32/27 reject;
        route 192.168.16.64/27 reject;
        route 192.168.16.96/27 reject;
        route 192.168.16.128/27 reject;
        route 192.168.16.160/27 reject;
        route 192.168.16.192/27 reject;
    }
    autonomous-system 1234;
}
protocols {
    bgp {
        group AS1000-Peers {
            type external;
            export send-statics;
            peer-as 1000;
            neighbor 10.222.70.2;
        }
    }
}
policy-options {
    policy-statement send-statics {
        term static-routes {
            from protocol static;
            then accept;
        }
    }
}
```

Requesting Specific Routes on the Customer 2 Router

Customer 2 has a link to the ISP, as well as a link to AS 8000. This customer has requested specific customer routes from the ISP, as well as from AS 8000.
Customer 2 wants to use the ISP for transit service to the Internet, and has requested a default route from the ISP.

```
[edit]
interfaces {
    so-0/0/1 {
        description "Connection to ISP Router 3";
        unit 0 {
            family inet {
                address 10.222.61.2/30;
            }
        }
    }
    so-0/0/2 {
        description "Connection to Private-Peer 2";
        unit 0 {
            family inet {
                address 10.222.6.1/30;
            }
        }
    }
    fxp0 {
        description "MGMT INTERFACE - DO NOT DELETE";
```

```
        unit 0 {
          family inet {
            address 10.251.0.8/24;
          }
        }
      }
      lo0 {
        unit 0 {
          family inet {
            address 192.168.64.1/32;
          }
        }
      }
    }
    routing-options {
      static {
        route 192.168.64.0/25 reject;
        route 192.168.64.128/25 reject;
        route 192.168.65.0/25 reject;
        route 192.168.66.0/25 reject;
        route 192.168.67.0/25 reject;
        route 192.168.65.128/25 reject;
        route 192.168.66.128/25 reject;
        route 192.168.67.128/25 reject;
      }
      autonomous-system 2468;
    }
    protocols {
      bgp {
        group External-Peers {
          type external;
          import inbound-routes;
          export outbound-routes;
          neighbor 10.222.61.1 {
            peer-as 1000;
          }
          neighbor 10.222.6.2 {
            peer-as 8000;
          }
        }
      }
    }
    policy-options {
      policy-statement outbound-routes {
        term statics {
          from protocol static;
          then accept;
        }
        term internal-bgp-routes {
          from {
            protocol bgp;
            as-path my-own-routes;
          }
          then accept;
        }
        term no-transit {
```

```
        then reject;
      }
    }
    policy-statement inbound-routes {
      term AS1000-primary {
        from {
          protocol bgp;
          as-path AS1000-routes;
        }
        then {
          local-preference 200;
          accept;
        }
      }
      term AS8000-backup {
        from {
          protocol bgp;
          as-path AS8000-routes;
        }
        then {
          local-preference 50;
          accept;
        }
      }
    }
    as-path my-own-routes "()";
    as-path AS1000-routes "1000 .*";
    as-path AS8000-routes "8000 .*";
}
```

Configuring a Peer Policy on ISP Router 3

On ISP Router 3, a separate policy is in place for each customer. The default route for Customer 1 is being sent by the **customer-1-peer** policy. This policy finds the **0.0.0.0/0** default route in **inet.0** and accepts it. The policy also rejects all other routes, thereby not sending all BGP routes on the ISP router. The **customer-2-peer** policy is for Customer 2 and contains the same policy terms, which also send the default route and no other transit BGP routes. The additional terms in the **customer-2-peer** policy send the ISP customer routes to Customer 2. Because there are local static routes on ISP router 3 that represent local customers, these routes are sent as well as all other internal (**192.168.0/17**) routes announced to the local router by the other ISP routers.

```
[edit]
routing-options {
  static { # simulate local customer routes
    route 192.168.72.0/22 reject;
    route 192.168.76.0/22 reject;
    route 192.168.80.0/22 reject;
    route 192.168.84.0/22 reject;
    route 192.168.88.0/22 reject;
    route 192.168.92.0/22 reject;
    route 192.168.72.0/21 reject;
    route 192.168.80.0/21 reject;
    route 192.168.88.0/21 reject;
  }
```

```
    generate { # install a default route if certain routes
      route 0.0.0.0/0 policy if-upstream-routes-exist; # from the exchange peers are
        advertised using BGP
    }
    autonomous-system 1000;
}
protocols {
    bgp {
      group Internal-Peers {
        type internal;
        local-address 192.168.0.3;
        export internal-peers;
        neighbor 192.168.0.1;
        neighbor 192.168.0.2;
      }
      group Customer-2 {
        type external;
        export customer-2-peer;
        peer-as 2468;
        neighbor 10.222.61.2;
      }
      group Customer-1 {
        type external;
        export customer-1-peer;
        peer-as 1234;
        neighbor 10.222.70.1;
      }
    }
    isis {
      level 1 disable;
      interface so-0/0/0.0;
      interface ge-0/1/0.0;
      interface lo0.0;
    }
}
policy-options {
    policy-statement internal-peers { # advertise local customer routes to peers
      term statics {
        from protocol static;
        then accept;
      }
      term next hop self { # set the BGP routes next hop to self for EBGP
        then { # routes advertised to IBGP peers
          next-hop self;
        }
      }
    }
    policy-statement if-upstream-routes-exist {
      term only-certain-contributing-routes {
        from { # allow either the 10.100.0.0/17 or the 10.101.0.0/27 route
          route-filter 10.100.0.0/17 exact; # route to activate the generated route
          route-filter 10.101.0.0/27 exact; # route to activate the generated route
        }
        then accept; # do not allow any other route to activate
      } # the generated route in the routing table
      term reject-all-other-routes {
```

```
        then reject;
      }
    }
    policy-statement customer-2--peer { # advertise customer routes to all peers
      term statics {
        from protocol static;
        then accept;
      }
      term-isp-and-customer routes { # advertise internal AS 1000 customer
        from { # to the customer
          protocol-bgp;
          route-filter 192.168.0.0/17 orlonger;
        }
        then accept;
      } # advertise just the default route to AS 2468
      term default-route {
        from {
          route-filter 0.0.0.0/exact;
        }
        then accept;
      }
      term reject-all-other-routes { # do not advertise any other routes
        then reject;
      }
    }
    policy-statement customer-1-peer {
      term default-route { # advertise just the default route to AS 1234
        from {
          route-filter 0.0.0.0/0 exact;
        }
        then accept;
      }
      term reject-all-other-routes { # do not advertise any other routes
        then reject;
      }
    }
  }
```

Configuring Private and Exchange Peers on ISP Router 1 and 2

ISP Router 1 and ISP Router 2 each have two policies configured: the **private-peers** policy and the **exchange-peers** policy. Because of their similar configurations, this example describes the configuration only for ISP Router 2.

On ISP Router 2, the **private-peers** policy sends the ISP customer routes to the Private Peer 2 router. The policy accepts all local static routes (local ISP Router 2 customers) and all BGP routes in the **192.168.0/17** range (advertised by other ISP routers). These two terms represent the ISP customer routes. The final term rejects all other routes, which includes the entire Internet routing table sent by the exchange peers. These routes do not need to be sent to Private Peer 2 for two reasons:

- The peer already maintains a connection to Exchange Peer 2 in our example, so the routes are redundant.

- The Private Peer wants customer routes only. The **private-peers** policy accomplishes this goal. The **exchange-peers** policy sends routes to the Exchange Peer 2 router.

In the example, only two routes need to be sent to Exchange Peer 2:

- The aggregate route that represents the AS 1000 routing space of **192.168.0/17**. This route is configured as an aggregate route locally and is advertised by the **exchange-peers** policy.
- The address space assigned to Customer 2, **192.168.64/22**. This smaller aggregate route needs to be sent to Exchange Peer 2 because the customer is also attached to the AS 8000 peer (Private Peer 2).

Sending these two routes to Exchange Peer 2 allows other networks in the Internet to reach the customer through either the ISP or the Private Peer. If just the Private Peer were to advertise the **/22** network while the ISP maintained only its **/17** aggregate, then all traffic destined for the customer would transit AS 8000 only. Because the customer also wants routes from the ISP, the **192.168.64/22** route is announced by ISP Router 2. Like the larger aggregate route, the **192.168.64/22** route is configured locally and is advertised by the **exchange-peers** policy. The final term in that policy rejects all routes, including the specific customer networks of the ISP, the customer routes from Private Peer 1, the customer routes from Private Peer 2, and the routing table from Exchange Peer 1. In essence, this final term prevents the ISP from performing transit services for the Internet at large.

```
[edit]
routing-options {
    static {
        route 192.168.32.0/22 reject;
        route 192.168.36.0/22 reject;
        route 192.168.40.0/22 reject;
        route 192.168.44.0/22 reject;
        route 192.168.48.0/22 reject;
        route 192.168.52.0/22 reject;
        route 192.168.32.0/21 reject;
        route 192.168.40.0/21 reject;
        route 192.168.48.0/21 reject;
    }
    aggregate {
        route 192.168.0.0/17;
        route 192.168.64.0/22;
    }
    autonomous-system 1000;
}
protocols {
    bgp {
        group Internal-Peers {
            type internal;
            local-address 192.168.0.2;
            export internal-peers;
            neighbor 192.168.0.1;
            neighbor 192.168.0.3;
        }
        group AS8000-Peers {
```

```
            type external;
            export private-peers;
            peer-as 8000;
            neighbor 10.222.45.2;
        }
        group AS22222-Peers {
            type external;
            export exchange-peers;
            peer-as 22222;
            neighbor 10.222.46.1;
        }
    }
    isis {
        level 1 disable;
        interface so-0/0/0.0;
        interface ge-0/2/0.0;
        interface lo0.0;
    }
}
policy-options {
    policy-statement internal-peers {
        term statics {
            from protocol static;
            then accept;
        }
        term next-hop-self {
            then {
                next-hop self;
            }
        }
    }
    policy-statement private-peers {
        term statics {
            from protocol static;
            then accept;
        }
        term isp-and-customer-routes {
            from {
                protocol bgp;
                route-filter 192.168.0.0/17 orlonger;
            }
            then accept;
        }
        term reject-all {
            then reject;
        }
    }
    policy-statement exchange-peers {
        term AS1000-Aggregate {
            from {
                protocol aggregate;
                route-filter 192.168.0.0/17 exact;
            }
            then accept;
        }
        term Customer-2-Aggregate {
```

```
        from {
          protocol aggregate;
          route-filter 192.168.64.0/22 exact;
        }
        then accept;
      }
      term reject-all-other-routes {
        then reject;
      }
    }
  }
```

Configuring Locally Defined Static Routes on the Exchange Peer 2 Router

The Exchange Peer 2 router exchanges all routes with all BGP peers. The **outbound-routes** policy for Exchange Peer 2 advertises locally defined static routes using BGP.

```
[edit]
protocols {
  bgp {
    group Peers {
      type external;
      export outbound-routes;
      neighbor 10.222.4.1 {
        peer-as 11111;
      }
      neighbor 10.222.44.2 {
        peer-as 8000;
      }
      neighbor 10.222.46.2 {
        peer-as 1000;
      }
    }
  }
}
policy-options {
  policy-statement outbound-routes { # advertise the simulated Internet routes
    term statics { # to all BGP peers
      from protocol static;
      then accept;
    }
  }
}
```

Configuring Outbound and Generated Routes on the Private Peer 2 Router

The Private Peer 2 router performs two main functions:

- Advertises routes local to AS 8000 to both the Exchange Peers and the ISP routers. The outbound-routes policy advertises the local static routes (that is, customers) on the router, and also advertises all routes learned by BGP that originated in either AS 8000 or AS 2468. These routes include other AS 8000 customer routes in addition to the AS 2468 customer. The AS routes are identified by an AS path regular expression match criteria in the policy.

- Advertises the **0.0.0.0/0** default route to the AS 2468 customer router. To accomplish this, the Private Peer creates a generated route for **0.0.0.0/0** locally on the router. This generated route is further assigned a policy called **if-upstream-routes-exist**, which allows only certain routes to contribute to the generated route, making it an active route in the routing table. Once the route is active, it can be sent to the AS 2468 router using BGP and the configured policies. The **if-upstream-routes-exist** policy accepts only the **20.100.0.0/17** route from Exchange Peer 2, and rejects all other routes. If the **20.100.0.0/17** route is withdrawn by the Exchange Peer, the Private Peer loses the **0.0.0.0/0** default route and withdraws the default route from the AS 2468 customer router.

```
[edit]
routing-options { # simulate local customer routes
  static {
    route 172.16.64.0/20 reject;
    route 172.16.80.0/20 reject;
    route 172.16.96.0/20 reject;
    route 172.16.112.0/20 reject;
    route 172.16.72.0/21 reject;
    route 172.16.88.0/21 reject;
    route 172.16.104.0/21 reject;
    route 172.16.120.0/21 reject;
  }
generate {
  route 0.0.0.0/0 policy if-upstream-routes-exist;
}
autonomous-system 8000;
protocols {
  bgp {
    group External-Peers {
      type external;
      export outbound-routes;
      neighbor 10.222.44.1 {
        peer-as 22222;
      }
      neighbor 10.222.45.1 {
        peer-as 1000;
      }
    }
    group Customers {
      type external;
      export internal-routes;
      neighbor 10.222.6.1 {
        peer-as 2468;
      }
    }
  }
}
policy-options {
  policy-statement outbound-routes { # advertise local customer routes
    term statics {
      from protocol static;
      then accept;
    }
    term allowed-bgp-routes {
```

```
        from { # advertise routes
        as-path [ my-own-routes AS2468-routes ];
        }
        then accept;
      }
      term no-transit {
        then reject; # do not advertise any other routes
      }
    }
    policy-statement internal-routes { # advertise local customer routes
      term statics {
        from protocol static;
        then accept;
      }
      term default-route { # advertise just the default route
        from {
          route-filter 0.0.0.0/0 exact;
        }
        then accept;
      }
      term reject-all-other-routes { # do not advertise any other routes
        then reject;
      }
    }
    policy-statement if-upstream-routes-exist {
      term as-22222-routes {
        from { # allow the 10.100.0.0/17 route to activate
          route-filter 10.100.0.0/17 exact; # the generated route in the routing # table
        }
        then accept;
      }
      term reject-all-other-routes {
        then reject; # do not allow any other route to activate
      } # the generated route in the routing table
    }
    as-path my-own-routes "()";
    as-path AS2468-routes "2468";
  }
```

CHAPTER 9

Extended Match Conditions Configuration

Configuring AS Path Regular Expressions to Use as Routing Policy Match Conditions

A BGP *AS path* is a path to a destination. An AS path consists of the AS numbers of networks that a packet traverses if it takes the associated route to a destination. The AS numbers are assembled in a sequence, or path, that is read from right to left. For example, for a packet to reach a destination using a route with an AS path **5 4 3 2 1**, the packet first traverses AS 1 and so on until it reaches AS 5, which is the last AS before its destination.

You can define a match condition based on all or portions of the AS path. To do this, you create a named AS path regular expression and then include it in a routing policy.

The following sections discuss the following tasks for configuring AS path regular expressions and provides the following examples:

Configuring AS Path Regular Expressions

You can create a named AS path regular expression and then include it in a routing policy with the **as-path** match condition (described in "Configuring Match Conditions in Routing Policy Terms" on page 47). To create a named AS path regular expression, include the **as-path** statement:

```
as-path name regular-expression;
```

You can include this statement at the following hierarchy levels:

- **[edit policy-options]**
- **[edit logical-systems *logical-system-name* policy-options]**

To include the AS path regular expression in a routing policy, include the **as-path** match condition in the **from** statement:

```
as-path name regular-expression;
policy-statement policy-name {
  term term-name {
    from {
      names;
    }
  }
}
```

Additionally, you can create a named AS path group made up of AS path regular expressions and then include it in a routing policy with the **as-path-group** match condition. To create a named AS path group, include the **as-path-group** statement:

```
as-path-group group-name  {
  name [ regular-expressions ];
}
```

You can include this statement at the following hierarchy levels:

- **[edit policy-options]**
- **[edit logical-systems *logical-system-name* policy-options]**

To include the AS path regular expressions within the AS path group in a routing policy, include the **as-path-group** match condition in the **from** statement:

```
as-path-group group-name {
  name [ regular-expressions ];
}
policy-statement policy-name {
  term term-name {
    from {
      as-path-group group-name;
    }
  }
}
```

NOTE: You cannot include both of the **as-path** and **as-path-group** statements in the same policy term.

NOTE: You can include the names of multiple AS path regular expressions in the **as-path** match condition in the **from** statement. If you do this, only one AS path regular expression needs to match for a match to occur. The AS path regular expression matching is effectively a logical OR operation.

The AS path name identifies the regular expression. It can contain letters, numbers, and hyphens (-), and can be up to 65,536 characters. To include spaces in the name, enclose the entire name in quotation marks (“ ”).

The regular expression is used to match all or portions of the AS path. It consists of two components, which you specify in the following format:

term <operator>

- ***term***—Identifies an AS. You can specify it in one of the following ways:
  - AS number—The entire AS number composes one term. You cannot reference individual characters within an AS number, which differs from regular expressions as defined in POSIX 1003.2.
  - Wildcard character—Matches any single AS number. The wildcard character is a period (.). You can specify multiple wildcard characters.
  - AS path—A single AS number or a group of AS numbers enclosed in parentheses. Grouping the regular expression in this way allows you to perform a common operation on the group as a whole and to give the group precedence. The grouped path can itself include operators.

    In Junos OS Release 9.1 and later, you can specify 4-byte AS numbers as defined in RFC 4893, *BGP Support for Four-octet AS Number Space*, as well as the 2-byte AS numbers that are supported in earlier releases of the Junos OS. You can configure a value in the range from 1 through 4,294,967,295.
- ***operator***—(Optional) An operator specifying how the term must match. Most operators describe how many times the term must be found to be considered a match (for example, any number of occurrences, or zero, or one occurrence). Table 10 on page 112 lists the regular expression operators supported for AS paths. You place operators immediately after ***term*** with no intervening space, except for the pipe (|) and dash (–) operators, which you place between two terms, and parentheses, with which you enclose terms.

You can specify one or more term–operator pairs in a single regular expression.

Table 11 on page 112 shows examples of how to define regular expressions to match AS paths.

Table 10: AS Path Regular Expression Operators

Operator	Match Definition
{*m*,*n*}	At least ***m*** and at most ***n*** repetitions of ***term***. Both ***m*** and ***n*** must be positive integers, and ***m*** must be smaller than ***n***.
{*m*}	Exactly ***m*** repetitions of ***term***. ***m*** must be a positive integer.
{*m*,}	***m*** or more repetitions of ***term***. ***m*** must be a positive integer.
*	Zero or more repetitions of ***term***. This is equivalent to **{0,}**.
+	One or more repetitions of ***term***. This is equivalent to **{1,}**.
?	Zero or one repetition of ***term***. This is equivalent to **{0,1}**.
|	One of two terms on either side of the pipe.
–	Between a starting and ending range, inclusive.
^	A character at the beginning of a community attribute regular expression. This character is added implicitly; therefore, the use of it is optional.
$	A character at the end of a community attribute regular expression. This character is added implicitly; therefore, the use of it is optional.
( )	A group of terms that are enclosed in the parentheses. Intervening space between the parentheses and the terms is ignored. If a set of parentheses is enclosed in quotation marks with no intervening space "()", it indicates a null path.
[ ]	Set of AS numbers. One AS number from the set must match. To specify the start and end of a range, use a hyphen (-). A carrot (^) may be used to indicate that it does not match a particular AS number in the set, for example [^123].

Table 11: Examples of AS Path Regular Expressions

AS Path to Match	Regular Expression	Sample Matches
AS path is 1234	1234	1234
Zero or more occurrences of AS number 1234	1234*	1234 1234 1234 1234 1234 1234 Null AS path
Zero or one occurrence of AS number 1234	1234? or 1234{0,1}	1234 Null AS path

Table 11: Examples of AS Path Regular Expressions *(continued)*

AS Path to Match	Regular Expression	Sample Matches
One through four occurrences of AS number 1234	1234{1,4}	1234 1234 1234 1234 1234 1234 1234 1234 1234 1234
One through four occurrences of AS number 12, followed by one occurrence of AS number 34	12{1,4} 34	12 34 12 12 34 12 12 12 34 12 12 12 12 34
Range of AS numbers to match a single AS number	123–125	123 124 125
	[123–125]*	Null AS path 123 124 124 125 125 125 123 124 125 123
Path whose second AS number must be 56 or 78	(. 56) | (. 78) or . (56 | 78)	1234 56 1234 78 9876 56 3857 78
Path whose second AS number might be 56 or 78	. (56 | 78)?	1234 56 52 34 56 1234 1234 78 39 794 78 2
Path whose first AS number is 123 and second AS number is either 56 or 78	123 (56|78)	123 56 123 78
Path of any length, except nonexistent, whose second AS number can be anything, including nonexistent	. .* or . .{0,}	1234123456781234567 8

Table 11: Examples of AS Path Regular Expressions *(continued)*

AS Path to Match	Regular Expression	Sample Matches
AS path is 1 2 3	1 2 3	1 2 3
One occurrence of the AS numbers 1 and 2, followed by one or more occurrences of the AS number 3	1 2 3+	1 2 3 1 2 3 3 1 2 3 3 3
One or more occurrences of AS number 1, followed by one or more occurrences of AS number 2, followed by one or more occurrences of AS number 3	1+ 2+ 3+	1 2 3 1 1 2 3 1 1 2 2 3 1 1 2 2 3 3
Path of any length that begins with AS numbers 4, 5, 6	4 5 6 .*	4 5 6 4 5 6 7 8 9
Path of any length that ends with AS numbers 4, 5, 6	.* 4 5 6	4 5 6 1 2 3 4 5 6 4 9 4 5 6
AS path 5, 12, or 18	5 | 12 | 18	5 12 18

Configuring a Null AS Path

You can use AS path regular expressions to create a null AS path that matches routes (prefixes) that have originated in your AS. These routes have not been advertised to your AS by any external peers. To create a null AS path, use the parentheses operator enclosed in quotation marks with no intervening spaces:

```
"()"
```

In the following example, locally administered AS 2 is connected to AS 1 (**10.2.2.6**) and AS 3. AS 3 advertises its routes to AS 2, but the administrator for AS 2 does not want to advertise AS 3 routes to AS 1 and thereby allow transit traffic from AS 1 to AS 3 through AS 2. To prevent transit traffic, the export policy **only-my-routes** is applied to AS 1. It permits advertisement of routes from AS 2 to AS 1 but prevents advertisement of routes for AS 3 (or routes for any other connected AS) to AS 1:

```
[edit policy-options]
null-as "()";
policy-statement only-my-routes {
   term just-my-as {
```

```
    from {
      protocol bgp;
      as-path null-as;
    }
    then accept;
  }
  term nothing-else {
    then reject;
  }
}
protocol {
  bgp {
    neighbor 10.2.2.6 {
      export only-my-routes;
    }
  }
}
```

How AS Path Regular Expressions Are Evaluated

AS path regular expressions implement the extended (modern) regular expressions as defined in POSIX 1003.2. They are identical to the UNIX regular expressions with the following exceptions:

- The basic unit of matching in an AS path regular expression is the AS number and not an individual character.
- A regular expression matches a route only if the AS path in the route exactly matches ***regular-expression***. The equivalent UNIX regular expression is **^*regular-expression*$**. For example, the AS path regular expression **1234** is equivalent to the UNIX regular expression **^1234$**.
- You can specify a regular expression using wildcard operators.

Examples: Configuring AS Path Regular Expressions

Exactly match routes with the AS path **1234 56 78 9** and accept them:

```
[edit]
policy-options {
  wellington "1234 56 78 9";
  policy-statement from-wellington {
    term term1 {
      from as-path wellington;
    }
      then {
        preference 200;
        accept;
      }
    term term2 {
      then reject;
    }
  }
}
```

Match alternate paths to an AS and accept them after modifying the preference:

```
[edit]
policy-options {
  wellington-alternate “1234{1,6} (56|47)? (78|101|112)* 9+”;
  policy-statement from-wellington {
    from as-path wellington-alternate;
    }
    then {
      preference 200;
      accept;
    }
  }
}
```

Match routes with an AS path of **123**, **124**, or **125** and accept them after modifying the preference:

```
[edit]
policy-options {
  addison "123–125";
  policy-statement from-addison {
    from as-path addison;
    }
    then {
      preference 200;
      accept;
    }
  }
}
```

Defining BGP Communities and Extended Communities for Use in Routing Policy Match Conditions

BGP communites can help you control routing policy. An example of a good use for BGP communities is unequal load balancing. When an autonomous system border router (ASBR) receives routes from directly connected external (EBGP) neighbors, the ASBR then advertises those routes to internal neighbors, using IBGP advertisements. In the IBGP adversisements, you can attach the link-bandwidth community to communicate the bandwidth of the advertised external link. This is useful when multiple external links are available, and you want to do unequal load balancing over the links. You configure the link-bandwidth extended community on all ingress links of the AS.

NOTE: The bandwidth information in the link-bandwidth extended community is based on the configured bandwidth of the EBGP link. It is not based on the amount of traffic on the link.

The Junos OS supports BGP link-bandwidth and multipath load balancing, as described in Internet draft draft-ietf-idr-link-bandwidth-01.txt, *BGP Link Bandwidth Extended Community* (expires August 2010).

To use a BGP community or extended community as a routing policy match condition, you define the community as described in the following sections:

Defining BGP Communities for Use in Routing Policy Match Conditions

To create a named BGP community and define the community members, include the **community** statement:

```
community name {
    invert-match;
    members [ community-ids ];
}
```

You can include this statement at the following hierarchy levels:

- **[edit policy-options]**
- **[edit logical-systems *logical-system-name* policy-options]**

name identifies the community. It can contain letters, numbers, and hyphens (-) and can be up to 255 characters long. To include spaces in the name, enclose the entire name in quotation marks (“ ”).

community-ids identifies one or more members of the community. Each community ID consists of two components, which you specify in the following format:

as-number:*community-value*;

- ***as-number***—AS number of the community member. It can be a value from 0 through 65,535. You can use the following notation in specifying the AS number:
  - String of digits.
  - Asterisk (*)—A wildcard character that matches all AS numbers. (In the definition of the community attribute, the asterisk also functions as described in Table 12 on page 119.)
  - Period (.)—A wildcard character that matches any single digit in an AS number.
  - Group of AS numbers—A single AS number or a group of AS numbers enclosed in parentheses. Grouping the numbers in this way allows you to perform a common operation on the group as a whole and to give the group precedence. The grouped numbers can themselves include regular expression operators. For more information about regular expressions, see “Using UNIX Regular Expressions in Community Names” on page 118.
- ***community-value***—Identifier of the community member. It can be a number from 0 through 65,535. You can use the following notation in specifying the community ID:

- String of digits.
- Asterisk (*)—A wildcard character that matches all community values. (In the definition of the community attribute, the asterisk also functions as described in Table 12 on page 119.)
- Period (.)—A wildcard character that matches any single digit in a community value number.
- Group of community value numbers—A single community value number or a group of community value numbers enclosed in parentheses. Grouping the regular expression in this way allows you to perform a common operation on the group as a whole and to give the group precedence. The grouped path can itself include regular expression operators.

You can also include one of the following well-known community names (defined in RFC 1997, *BGP Communities Attribute*) in the ***community-ids*** option for the **members** statement:

- **no-advertise**—Routes in this community name must not be advertised to other BGP peers.
- **no-export**—Routes in this community must not be advertised outside a BGP confederation boundary.
- **no-export-subconfed**—Routes in this community must not be advertised to external BGP peers, including peers in other members' ASs inside a BGP confederation.

Using UNIX Regular Expressions in Community Names

When specifying the members of a named BGP community (in the **members [** ***community-ids*** **]** statement), you can use UNIX-style regular expressions to specify the AS number and the member identifier. A regular expression consists of two components, which you specify in the following format:

term operator;

term identifies the string to match.

operator specifies how the term must match. Table 12 on page 119 lists the regular expression operators supported in community IDs. You place an operator immediately after ***term*** with no intervening space, except for the pipe (**|**) and dash (**–**) operators, which you place between two terms, and parentheses, with which you enclose terms. Table 13 on page 120 shows examples of how to define ***community-ids*** using community regular expressions. The operator is optional.

Community regular expressions are identical to the UNIX regular expressions. Both implement the extended (or modern) regular expressions as defined in POSIX 1003.2.

Community regular expressions evaluate the string specified in ***term*** on a character-by-character basis. For example, if you specify **1234:5678** as ***term***, the regular expressions see nine discrete characters, including the colon (**:**), instead of two sets of numbers (**1234** and **5678**) separated by a colon.

NOTE: In Junos OS Release 9.1 and later, you can specify 4-byte AS numbers as defined in RFC 4893, *BGP Support for Four-octet AS Number Space*, as well as the 2-byte AS numbers that are supported in earlier releases of the Junos OS.

Table 12: Community Attribute Regular Expression Operators

Operator	Match Definition
{*m*,*n*}	At least ***m*** and at most ***n*** repetitions of ***term***. Both ***m*** and ***n*** must be positive integers, and ***m*** must be smaller than ***n***.
{*m*}	Exactly ***m*** repetitions of ***term***. ***m*** must be a positive integer.
{*m*,}	***m*** or more repetitions of ***term***. ***m*** must be a positive integer.
*	Zero or more repetitions of ***term***. This is equivalent to **{0,}**.
+	One or more repetitions of ***term***. This is equivalent to **{1,}**.
?	Zero or one repetition of ***term***. This is equivalent to **{0,1}**.
|	One of the two terms on either side of the pipe.
–	Between a starting and ending range, inclusive.
^	Character at the beginning of a community attribute regular expression. We recommend the use of this operator for the clearest interpretation of your community attribute regular expression. If you do not use this operator, the regular expression **123:456** could also match a route tagged with **5123:456**.
$	Character at the end of a community attribute regular expression. We recommend the use of this operator for the clearest interpretation of your community attribute regular expression. If you do not use this operator, the regular expression **123:456** could also match a route tagged with **123:4563**.
[ ]	Set of characters. One character from the set can match. To specify the start and end of a range, use a hyphen (-). To specify a set of characters that do not match, use the caret (^) as the first character after the opening square bracket (**[**).
( )	Group of terms that are enclosed in parentheses. If enclosed in quotation marks with no intervening space ("**()**"), indicates a null. Intervening space between the parentheses and the terms is ignored.
“ ”	Characters (such as space, tab, question mark, and bracket) that are enclosed within quotation marks in a community attribute regular expression indicate special characters.

Table 13: Examples of Community Attribute Regular Expressions

Community Attribute to Match	Regular Expression	Sample Matches
AS number is 56 or 78. Community value is any number.	^((56) | (78)):(.*)$	56:1000 78:65000
AS number is 56. Community value is any number that starts with 2.	^56:(2.*)$	56:2 56:222 56:234
AS number is any number. Community value is any number that ends with 5, 7, or 9.	^(.*):(.*[579])$	1234:5 78:2357 34:65009
AS number is 56 or 78. Community value is any number that starts with 2 and ends with 2 through 8.	^((56) | (78)):(2.*[2–8])$	56:22 56:21197 78:2678

Defining BGP Extended Communities for Use in Routing Policy Match Conditions

To create a named BGP community and define the community members, include the **community** statement:

```
community name {
    invert-match;
    members [ community-ids ];
}
```

You can include this statement at the following hierarchy levels:

- **[edit policy-options]**
- **[edit logical-systems *logical-system-name* policy-options]**

name identifies the community. It can contain letters, numbers, and hyphens (-) and can be up to 255 characters long. To include spaces in the name, enclose the entire name in quotation marks (“ ”).

community-ids identifies one or more members of the community. Each community ID consists of three components, which you specify in the following format:

type:*administrator*:*assigned-number*

type is the type of extended community and can be either the 16-bit numerical identifier of a specific BGP extended community or one of these types:

- **bandwidth**—Sets up the bandwidth extended community. Specifying link bandwidth allows you to distribute traffic unequally among different BGP paths.

  NOTE: The link bandwidth attribute does not work concurrently with per-prefix load balancing.

- **domain-id**—Identifies the OSPF domain from which the route originated.
- **origin**—Identifies where the route originated.
- **rt-import**—Identifies the route to install in the routing table.

  NOTE: You must identify the route by an IP address, not an AS number.

- **src-as**—Identifies the AS from which the route originated. You must specify an AS number, not an IP address.

  NOTE: You must identify the AS by an AS number, not an IP address.

- **target**—Identifies the destination to which the route is going.

  NOTE: For an import policy for a VPN routing and forwarding (VRF) instance, you must include at least one route target. Additionally, you cannot use wildcard characters or regular expressions in the route target for a VRF import policy. Each value you configure for a route target for a VRF import policy must be a single value.

administrator is the administrator. It is either an AS number or an IP version 4 (IPv4) address prefix, depending on the type of extended community.

assigned-number identifies the local provider.

In Junos OS Release 9.1 and later, you can specify 4-byte AS numbers as defined in RFC 4893, *BGP Support for Four-octet AS Number Space*, as well as the 2-byte AS numbers that are supported in earlier releases of the Junos OS. In plain-number format, you can configure a value in the range from 1 through 4,294,967,295. To configure a **target** or **origin** extended community that includes a 4-byte AS number in the plain-number format, append the letter "L" to the end of number. For example, a target community with the 4-byte AS number 334,324 and an assigned number of 132 is represented as **target:334324L:132**.

In Junos OS Release 9.2 and later, you can also use AS-dot notation when defining a 4-byte AS number for the **target** and **origin** extended communities. Specify two integers

joined by a period: *16-bit high-order value in decimal.16-bit low-order value in decimal*. For example, the 4-byte AS number represented in plain-number format as 65546 is represented in AS-dot notation as 1.10.

Examples: Defining BGP Extended Communities

Configure a target community with an administrative field of **10458** and an assigned number of **20**:

```
[edit policy-options]
community test-a members [ target:10458:20 ];
```

Configure a target community with an administrative field of **10.1.1.1** and an assigned number of **20**:

```
[edit policy-options]
community test-a members [ target:10.1.1.1:20 ];
```

Configure an origin community with an administrative field of **10.1.1.1** and an assigned number of **20**:

```
[edit policy-options]
community test-a members [ origin:10.1.1.1:20 ];
```

Configure a target community with a 4-byte AS number in the administrative field of **100000** and an assigned number of **130**:

```
[edit policy-options]
community test-b members [ target:100000L:130 ];
```

Related Documentation

- *Using 4-Byte Autonomous System Numbers in BGP Networks Technology Overview*
- Configuring 4-Byte AS Numbers and BGP Extended Community Attributes in the *Using 4-Byte Autonomous System Numbers in BGP Networks Technology Overview*

Inverting Community Matches

To invert the results of the community expression matching, include the **invert-match** statement:

```
invert-match;
```

You can include this statement at the following hierarchy levels:

- **[edit policy-options** community ***name*****]**
- **[edit logical-systems** ***logical-system-name*** **policy-options** community ***name*****]**

Including BGP Communities and Extended Communities in Routing Policy Match Conditions

To include a BGP community or extended community in a routing policy match condition, include the **community** condition in the **from** statement of a policy term:

```
from {
```

```
    community [ names ];
}
```

You can include this statement at the following hierarchy levels:

- [edit policy-options policy-statement *policy-name* term *term-name*]
- [edit logical-systems *logical-system-name* policy-options policy-statement *policy-name* term *term-name*]

Additionally, you can explicitly exclude BGP community information with a static route by using the **none** option. Include this option when configuring an individual route in the **route** portion to override a community option specified in the **defaults** portion.

NOTE: You can include the names of multiple communities in the community match condition. If you do this, only one community needs to match for a match to occur (matching is effectively a logical OR operation).

Using Routing Policies to Prevent Advertisement of BGP Communities to Neighbors

By default, communities are sent to BGP peers. To suppress the advertisement of communities to a neighbor, remove all communities. When the result of an export policy is an empty set of communities, the community attribute is not sent. To remove all communities, first define a wildcard set of communities (here, the community is named **wild**):

```
[edit policy-options]
community wild members "* : *";
```

Then, in the routing policy statement, specify the **community delete** action:

```
[edit policy-options]
policy-statement policy-name {
    term term-name {
        then community delete wild;
    }
}
```

To suppress a particular community from any AS, define the community as **community wild members "*:*community-value*"**.

Examples: Configuring BGP Communities as Routing Policy Match Conditions

Create a community named **dunedin** and apply it in a routing policy statement:

```
[edit]
policy-options {
    community dunedin members [ 56:2379 23:46944 ];
    policy-statement from-dunedin {
        from community dunedin;
        then {
            metric 2;
            preference 100;
```

```
        next policy;
      }
    }
  }
```

The preceding example modifies the metric and preference for routes that contain members of community **dunedin** only.

NOTE: You cannot set or add a community in a policy whose members use regular expressions or a wildcard.

Delete a particular community from a route, leaving remaining communities untouched:

```
[edit]
policy-options {
  community dunedin members 701:555;
  policy-statement delete-dunedin {
    then {
      community delete dunedin;
    }
  }
}
```

Remove any community from a route with the AS number of **65534** or **65535**:

```
[edit]
policy-options {
  community my-as1-transit members [ 65535:10 65535:11 ];
  community my-as2-transit members [ 65534:10 65534:11 ];
  community my-wild members [ 65534:* 65535:* ];
  policy-statement delete-communities {
    from {
      community [ my-as1-transit my-as2-transit ];
    }
    then {
      community delete my-wild;
    }
  }
}
```

Match the set of community members 5000, 5010, 5020, 5030, and so on up to 5090:

```
[edit]
policy-options {
  community customers members "^1111:50.0$";
  policy-statement advertise-customers {
    from community customers;
    then accept;
  }
}
```

Reject routes that are longer than **/19** in Class A space, **/16** in Class B space, and **/24** in Class C space:

```
[edit policy-options]
```

```
community auckland-accept members 555:1;
policy-statement drop-specific-routes {
  from {
    route-filter 0.0.0.0/1 upto /19 {
      community add auckland-accept;
      next policy;
    }
    route-filter 172.16.0.0/2 upto /16 {
      community add auckland-accept;
      next policy;
    }
    route-filter 192.168.0.0/3 upto /24 {
      community add auckland-accept;
      next policy;
    }
  }
  then reject;
}
```

In the preceding example, for routes that are not rejected, the tag **auckland-accept** is added.

Create routing policies to handle peer and customer communities. This example does the following:

- Customer routes that match the attributes defined in the **lcl20*x*-low** communities, for example, **lcl201-low**, are accepted and their local preference is changed to 80.
- Customer routes that match the attributes defined in the **lcl20*x*-high** communities, for example, **lcl201-high**, are accepted and have their local preference changed to 120.
- Internal routes that match the attributes defined in the **internal20*x*** communities, for example, **internal201**, are rejected and not advertised to customers.
- Routes received from a peer are assigned a metric of 10 and the community defined in **peer201**.
- Routes that match the attributes defined in the **prepend20*x*-*x*** communities, for example, **prepend201-1**, **prepend201-2**, or **prepend201-3**, are sent to peers and have the AS number 201 prepended the specified number of times.
- Routes that match the attributes defined in the **peer20*x***, **custpeer20*x***, and **internal20*x*** communities, for example, **peer201**, **custpeer201**, or **internal201**, respectively, are rejected and not advertised to peers.

```
[edit]
policy-options {
  community internal201 members 201:112;
  community internal202 members 202:112;
  community internal203 members 203:112;
  community internal204 members 204:112;
  community internal205 members 205:112;
  community peer201 members 201:555;
  community peer202 members 202:555;
  community peer203 members 203:555;
  community peer204 members 204:555;
  community peer205 members 205:555;
```

```
community custpeer201 members 201:20;
community custpeer202 members 202:20;
community custpeer203 members 203:20;
community custpeer204 members 204:20;
community custpeer205 members 205:20;
community prepend201-1 members 201:1;
community prepend202-1 members 202:1;
community prepend203-1 members 203:1;
community prepend204-1 members 204:1;
community prepend205-1 members 205:1;
community prepend201-2 members 201:2;
community prepend202-2 members 202:2;
community prepend203-2 members 203:2;
community prepend204-2 members 204:2;
community prepend205-2 members 205:2;
community prepend201-3 members 201:3;
community prepend202-3 members 202:3;
community prepend203-3 members 203:3;
community prepend204-3 members 204:3;
community prepend205-3 members 205:3;
community lcl201-low members 201:80;
community lcl202-low members 202:80;
community lcl203-low members 203:80;
community lcl204-low members 204:80;
community lcl205-low members 205:80;
community lcl20x-high members "^20 [ 1-5 ] : 120$";
policy-statement in-customer {
  term term1 {
    from {
      protocol bgp;
      community lcl20x-high;
    }
    then {
      local-preference 80;
      accept;
    }
  }
  term term2 {
    from {
      protocol bgp;
      community [ lcl201-high lcl202-high lcl203-high lcl204-high lcl205-high ];
    }
    then local-preference 120;
  }
  then next policy;
}
policy-statement out-customer {
  term term1 {
    from {
      protocol bgp;
      community [internal201 internal202 internal203 internal204 internal205];
    }
    then reject;
  }
  then next policy;
}
```

```
    policy-statement in-peer {
      from protocol bgp;
      then {
        metric 10;
        community set peer201;
      }
    }
    policy-statement out-peer {
      term term1 {
        from {
          protocol bgp;
          community [ prepend201-1 prepend202-1 prepend203-1 prepend204-1
            prepend205-1 ];
        }
        then as-path-prepend 201;
      }
      term term2 {
        from {
          protocol bgp;
          community [ prepend201-2 prepend202-2 prepend203-2 prepend204-2
            prepend205-2 ];
        }
        then as-path-prepend "201 201";
      }
      term term3 {
        from {
          protocol bgp;
          community [ prepend201-3 prepend202-3 prepend203-3 prepend204-3
            prepend205-3 ];
        }
        then as-path-prepend "201 201 201";
      }
      term term4 {
        from {
          protocol bgp;
          community [ peer201 peer202 peer203 peer204 peer205 custpeer201
            custpeer202 custpeer203 custpeer204 custpeer205 internal201 internal202
            internal203 internal204 internal205 ];
        }
        then reject;
      }
      then next policy;
    }
  }
```

Configuring Prefix Lists for Use in Routing Policy Match Conditions

A *prefix list* is a named list of IP addresses. You can specify an exact match with incoming routes and apply a common action to all matching prefixes in the list.

NOTE: Because the configuration of prefix lists includes setting up prefixes and prefix lengths, we strongly recommend that you have a thorough understanding of IP addressing, including supernetting, before proceeding with the configuration.

A prefix list functions like a route list that contains multiple instances of the **exact** match type only. The differences between these two extended match conditions are summarized in Table 14 on page 128.

Table 14: Prefix List and Route List Differences

Feature	Prefix List	Route Lists
Action	Can specify action in a **then** statement only. These actions are applied to all prefixes that match the term.	Can specify action that is applied to a particular prefix in a **route-filter** match condition in a **from** statement, or to all prefixes in the list using a **then** statement.

For information about configuring route lists, see "Configuring Route Lists for Use in Routing Policy Match Conditions" on page 131.

This section includes the following information:

Configuring Prefix Lists

You can create a named prefix list and include it in a routing policy with the **prefix-list** match condition (described in "Configuring Match Conditions in Routing Policy Terms" on page 47).

To define a prefix list, include the **prefix-list** statement:

```
prefix-list prefix-list-name {
    apply-path path;
    ip-addresses;
}
```

You can include this statement at the following hierarchy levels:

- **[edit policy-options]**
- **[edit logical-systems *logical-system-name* policy-options]**

You can use the **apply-path** statement to include all prefixes pointed to by a defined path, or you can specify one or more addresses, or both.

To include a prefix list in a routing policy, specify the **prefix-list** match condition in the **from** statement at the **[edit policy-options policy-statement** *policy-name* **term** *term-name***]** hierarchy level:

```
[edit policy-options policy-statement policy-name term term-name]
from {
  prefix-list prefix-list-name;
}
then actions;
```

name identifies the prefix list. It can contain letters, numbers, and hyphens (-) and can be up to 255 characters long. To include spaces in the name, enclose the entire name in quotation marks (“ ”).

ip-addresses are the IPv4 or IP version 6 (IPv6) prefixes specified as ***prefix/prefix-length***. If you omit ***prefix-length*** for an IPv4 prefix, the default is **/32**. If you omit ***prefix-length*** for an IPv6 prefix, the default is **/128**. Prefixes specified in a **from** statement must be either all IPv4 addresses or all IPv6 addresses.

NOTE: **You cannot apply actions to individual prefixes in the list.**

You can specify the same prefix list in the **from** statement of multiple routing policies or firewall filters. For information about firewall filters, see Guidelines for Configuring Standard Firewall Filters and Guidelines for Applying Standard Firewall Filters.

Use the **apply-path** statement to configure a prefix list comprising all IP prefixes pointed to by a defined path. This eliminates most of the effort required to maintain a group prefix list.

The path consists of elements separated by spaces. Each element matches a configuration keyword or an identifier, and you can use wildcards to match more than one identifier. Wildcards must be enclosed in angle brackets, for example, <*>.

NOTE: **You cannot add a path element, including wildcards, after a leaf statement in the apply-path statement. Path elements, including wildcards, can only be used after a container statement.**

NOTE: **When you use apply-path to define a prefix list, you can also use the same prefix list in a policy statement.**

For examples of configuring a prefix list, see “Example: Configuring a Prefix List” on page 131.

How Prefix Lists Are Evaluated in Routing Policy Match Conditions

During prefix list evaluation, the policy framework software performs a *longest-match lookup*, which means that the software searches for the prefix in the list with the longest

length. The order in which you specify the prefixes, from top to bottom, does not matter. The software then compares a route's source address to the longest prefix.

You can use prefix list qualifiers for prefixes contained in a prefix list by configuring a prefix list filter. For more information, see *Configuring Prefix Lists for Use in Routing Policy Match Conditions*.

If a match occurs, the evaluation of the current term continues. If a match does not occur, the evaluation of the current term ends.

NOTE: If you specify multiple prefixes in the prefix list, only one prefix must match for a match to occur. The prefix list matching is effectively a logical OR operation.

Configuring Prefix List Filters

A prefix list filter allows you to apply prefix list qualifiers to a list of prefixes within a prefix list. The prefixes within the list are evaluated using the specified qualifiers. You can configure multiple prefix list filters under the same policy term.

To configure a prefix list filter, include the **prefix-list-filter** statement at the **[edit policy-options policy-statement *policy-name* from]** hierarchy level:

```
[edit policy-options policy-statement policy-name]
from {
  prefix-list-filter prefix-list-name match-type actions;
}
```

The ***prefix-list-name*** option is the name of the prefix list to be used for evaluation. You can specify only one prefix list.

The ***match-type*** option is the type of match to apply to the prefixes in the prefix list. It can be one of the match types listed in Table 15 on page 130.

The ***actions*** option is the action to take if the prefix list matches. It can be one or more of the actions listed in "Configuring Flow Control Actions" on page 55 and "Configuring Actions That Manipulate Route Characteristics" on page 56.

Table 15: Route List Match Types for a Prefix List Filter

Match Type	Match Condition
exact	The route shares the same most-significant bits (described by ***prefix-length***), and ***prefix-length*** is equal to the route's prefix length.
longer	The route shares the same most-significant bits (described by ***prefix-length***), and ***prefix-length*** is greater than the route's prefix length.
orlonger	The route shares the same most-significant bits (described by ***prefix-length***), and ***prefix-length*** is equal to or greater than the route's prefix length.

Example: Configuring a Prefix List

The following example accepts and rejects traffic from sites specified using prefix lists:

```
[edit]
policy-options {
   policy-statement prefix-list-policy {
      term ok-sites {
         from {
            prefix-list known-ok-sites;
         }
         then accept;
      }
      term reject-bcasts {
         from {
            prefix-list known-dir-bcast-sites;
         }
         then reject;
      }
   }
}
[edit]
policy-options {
   prefix-list known-ok-sites {
      172.16.0.3;
      10.10.0.0/16;
      192.168.12.0/24;
   }
   [edit]
   policy-options {
      prefix-list known-dir-bcast-sites {
         10.3.4.6;
         10.2.0.0/16;
         192.168.1.0/24;
      }
   }
}
```

Configuring Route Lists for Use in Routing Policy Match Conditions

A *route list* is a collection of match prefixes. When specifying a match prefix, you can specify an exact match with a particular route or a less precise match. You can configure either a common action that applies to the entire list or an action associated with each prefix.

NOTE: Because the configuration of route lists includes setting up prefixes and prefix lengths, we strongly recommend that you have a thorough understanding of IP addressing, including supernetting, before proceeding with the configuration.

It is also important to understand how a route list is evaluated, particularly if the route list includes multiple **route-filter** options in a **from** statement. We strongly recommend that you read "How Route Lists Are Evaluated in Routing Policy Match Conditions" on page 136 before proceeding with the configuration. Not fully understanding the evaluation process could result in faulty configuration and unexpected results.

This section discusses the following topics:

Configuring Route Lists

To configure a route list, include one or more **route-filter** or **source-address-filter** statements:

```
route-filter destination-prefix match-type {
  actions;
}
source-address-filter source-prefix match-type {
  actions;
}
```

You can include these statements at the following hierarchy levels:

- **[edit policy-options policy-statement** *policy-name* **term** *term-name* **from]**
- **[edit logical-systems** *logical-system-name* **policy-options policy-statement** *policy-name* **term** *term-name* **from]**

The **route-filter** option is typically used to match an incoming route address to destination match prefixes of any type except for unicast source addresses.

The ***destination-prefix*** address is the IP version 4 (IPv4) or IP version 6 (IPv6) address prefix specified as ***prefix/prefix-length***. If you omit ***prefix-length*** for an IPv4 prefix, the default is **/32**. If you omit ***prefix-length*** for an IPv6 prefix, the default is **/128**. Prefixes specified in a **from** statement must be either all IPv4 addresses or all IPv6 addresses.

The **source-address-filter** option is typically used to match an incoming route address to unicast source addresses in multiprotocol BGP (MBGP) and Multicast Source Discovery Protocol (MSDP) environments.

source-prefix address is the IPv4 or IPv6 address prefix specified as ***prefix/prefix-length***. If you omit ***prefix-length*** for an IPv4 prefix, the default is **/32**. If you omit ***prefix-length*** for

an IPv6 prefix, the default is **/128**. Prefixes specified in a **from** statement must be either all IPv4 addresses or all IPv6 addresses.

match-type is the type of match to apply to the source or destination prefix. It can be one of the match types listed in Table 16 on page 134. For examples of the match types and the results when presented with various routes, see Table 17 on page 135.

actions are the actions to take if a route address matches the criteria specified for a destination match prefix (specified as part of a **route-filter** option) or for a source match prefix (specified as part of a **destination-address-filter** option). The actions can consist of one or more of the actions described in the following sections of "Configuring Actions in Routing Policy Terms" on page 54:

- Configuring Flow Control Actions on page 55
- Configuring Actions That Manipulate Route Characteristics on page 56

In a route list you can specify actions in two ways:

- In the **route-filter** or **source-address-filter** option—These actions are taken immediately after a match occurs, and the **then** statement is not evaluated.
- In the **then** statement—These actions are taken after a match occurs but no actions are specified for the **route-filter** or **source-address-filter** option.

The **upto** and **prefix-length-range** match types are similar in that both specify the most-significant bits and provide a range of prefix lengths that can match. The difference is that **upto** allows you to specify an upper limit only for the prefix length range, whereas **prefix-length-range** allows you to specify both lower and upper limits.

For more examples of these route list match types, see "Route List Examples" on page 138.

Table 16: Route List Match Types for a Prefix List

Match Type	Match Criteria
address-mask ***netmask-value***	All of the following are true: • The bit-wise logical AND of the ***netmask-value*** pattern and the incoming IPv4 route address and the bit-wise logical AND of the ***netmask-value*** pattern and the ***destination-prefix*** address are the same. The bits set in the ***netmask-value*** pattern do not need to be contiguous. • The ***prefix-length*** component of the incoming IPv4 route address and the ***prefix-length*** component of the ***destination-prefix*** address are the same. NOTE: The **address-mask** routing policy match type is valid only for matching an incoming IPv4 (**family inet**) route address to a list of destination match prefixes specified in a **route-filter** statement. The **address-mask** routing policy match type enables you to match an incoming IPv4 route address on a configured netmask address in addition to the length of a configured destination match prefix. The length of the route address must match exactly with the length of the configured destination match prefix, as the **address-mask** match type does not support prefix length variations for a range of prefix lengths. When the longest-match lookup is performed on a route list, the lookup evaluates an **address-mask** match type differently from other routing policy match types. The lookup does not consider the length of the destination match prefix. Instead, the lookup considers the number of contiguous high-order bits set in the netmask value. For more information about this route list match type, see "How an Address Mask Match Type Is Evaluated" on page 137. For example configurations showing route lists that contain the **address-mask** match type, see the following topics: • "Example: Accepting Incoming IPv4 Routes by Applying an Address Mask to the Route Address and the Destination Match Prefix" on page 143. • "Example: Accepting Incoming IPv4 Routes with Similar Patterns But Different Prefix Lengths" on page 144. • "Example: Evaluation of an Address Mask Match Type with Longest-Match Lookup" on page 145.
exact	All of the following are true: • The route address shares the same most-significant bits as the match prefix (***destination-prefix*** or ***source-prefix***). The number of significant bits is described by the ***prefix-length*** component of the match prefix. • The ***prefix-length*** component of the match prefix is equal to the route's prefix length.
longer	All of the following are true: • The route address shares the same most-significant bits as the match prefix (***destination-prefix*** or ***source-prefix***). The number of significant bits is described by the ***prefix-length*** component of the match prefix. • The route's prefix length is greater than the ***prefix-length*** component of the match prefix.
orlonger	All of the following are true: • The route address shares the same most-significant bits as the match prefix (***destination-prefix*** or the ***source-prefix***). The number of significant bits is described by the ***prefix-length*** component of the match prefix. • The route's prefix length is equal to or greater than the ***prefix-length*** component of the configured match prefix.

Table 16: Route List Match Types for a Prefix List *(continued)*

Match Type	Match Criteria
prefix-length-range *prefix-length2-prefix-length3*	All of the following are true: • The route address shares the same most-significant bits as the match prefix (***destination-prefix*** or ***source-prefix***). The number of significant bits is described by the ***prefix-length*** component of the match prefix. • The route's prefix length falls between ***prefix-length2*** and ***prefix-length3***, inclusive.
through {*destination-prefix2* | *source-prefix2*}	All of the following are true: • The route address shares the same most-significant bits as the first match prefix (***destination-prefix*** or ***source-prefix***). The number of significant bits is described by the ***prefix-length*** component of the first match prefix. • The route address shares the same most-significant bits as the second match prefix (***destination-prefix2*** or ***source-prefix2***). The number of significant bits is described by the ***prefix-length*** component of the second match prefix. • The route's prefix length is less than or equal to the ***prefix-length*** component of the second match prefix. You do not use the **through** match type in most routing policy configurations. For an example, see "Example: Rejecting Routes from Specific Hosts" on page 140.
upto *prefix-length2*	All of the following are true: • The route address shares the same most-significant bits as the match prefix (***destination-prefix*** or ***source-prefix***). The number of significant bits is described by the ***prefix-length*** component of the match prefix. • The route's prefix length falls between the ***prefix-length*** component of the first match prefix and ***prefix-length2***.

Table 17: Match Type Examples

Prefix	192.168/16 exact	192.168/16 longer	192.168/16 orlonger	192.168/16 upto /24	192.168/16 prefix-length-range /18 – /20	192.168/16 through 192.168.16/20	192.168/19 address-mask 255.255.0.0
10.0.0.0/8	–	–	–	–	–	–	–
192.168.0.0/16	Match	–	Match	Match	–	Match	–
192.168.0.0/17	–	Match	Match	Match	–	Match	–
192.168.0.0/18	–	Match	Match	Match	Match	Match	–
192.168.0.0/19	–	Match	Match	Match	Match	Match	Match
192.168.4.0/24	–	Match	Match	Match	–	–	–
192.168.5.4/30	–	Match	Match	–	–	–	–
192.168.12.4/30	–	Match	Match	–	–	–	–

Table 17: Match Type Examples *(continued)*

Prefix	192.168/16 exact	192.168/16 longer	192.168/16 orlonger	192.168/16 upto /24	192.168/16 prefix-length-range /18 – /20	192.168/16 through 192.168.16/20	192.168/19 address-mask 255.255.0.0
192.168.12.128/32	–	Match	Match	–	–	–	–
192.168.16.0/20	–	Match	Match	Match	Match	Match	–
192.168.192.0/18	–	Match	Match	Match	Match	–	–
192.168.224.0/19	–	Match	Match	Match	Match	–	Match
10.169.1.0/24	–	–	–	–	–	–	–
10.170.0.0/16	–	–	–	–	–	–	–

How Route Lists Are Evaluated in Routing Policy Match Conditions

During route list evaluation, the policy framework software compares each route's source address with the destination prefixes in the route list. The evaluation occurs in two steps:

1. The policy framework software performs a *longest-match lookup*, which means that the software searches for the prefix in the list with the longest length.

   The longest-match lookup considers the ***prefix*** and ***prefix-length*** components of the configured match prefix only, and not the ***match-type*** component. The following sample route list illustrates this point:

   ```
   from {
     route-filter 192.168.0.0/14 upto /24 reject;
     route-filter 192.168.0.0/15 exact;
   }
   then accept;
   ```

   The longest match for the candidate route **192.168.1.0/24** is the second route-filter, **192.168.0.0/15**, which is based on prefix and prefix length only.

2. Once an incoming route matches a prefix (longest first), the following actions occur:

   - The route filter stops evaluating other prefixes, even if the match type fails.
   - The software examines the match type and action associated with that prefix.

NOTE: When a route source address is evaluated against a match criteria that uses the address-mask match type, both steps of the evaluation include the configured netmask value. For more information, see "How an Address Mask Match Type Is Evaluated" on page 137.

In Step 1, if route **192.168.1.0/24** were evaluated, it would fail to match. It matches the longest prefix of **192.168.0.0/15**, but it does not match **exact**. The route filter is finished because it matched a prefix, but the result is a failed match because the match type failed.

If a match occurs, the action specified with the prefix is taken. If an action is not specified with the prefix, the action in the **then** statement is taken. If neither action is specified, the software evaluates the next term or routing policy, if present, or takes the **accept** or **reject** action specified by the default policy. For more information about the default routing policies, see "Default Routing Policies and Actions" on page 16.

NOTE: **If you specify multiple prefixes in the route list, only one prefix needs to match for a match to occur. The route list matching is effectively a logical OR operation.**

If a match does not occur, the software evaluates the next term or routing policy, if present, or takes the **accept** or **reject** action specified by the default policy.

For example, compare the prefix **192.168.254.0/24** against the following route list:

```
route-filter 192.168.0.0/16 orlonger;
route-filter 192.168.254.0/23 exact;
```

The prefix **192.168.254.0/23** is determined to be the longest prefix. When the software evaluates **192.168.254.0/24** against the longest prefix, a match occurs (**192.168.254.0/24** is a subset of **192.168.254.0/23**). Because of the match between **192.168.254.0/24** and the longest prefix, the evaluation continues. However, when the software evaluates the match type, a match does not occur between **192.168.254.0/24** and **192.168.254.0/23 exact**. The software concludes that the term does not match and goes on to the next term or routing policy, if present, or takes the **accept** or **reject** action specified by the default policy.

How an Address Mask Match Type Is Evaluated

The **address-mask** routing policy match type enables you to match incoming IPv4 route addresses on a configured netmask value in addition to the length of a configured destination match prefix. During route list evaluation, an **address-mask** match type is processed differently from other routing policy match types, taking into consideration the configured netmask value:

1. When a longest-match lookup evaluates an **address-mask** routing policy match type, the ***prefix-length*** component of the configured match prefix is not considered. Instead, the lookup considers the number of contiguous high-order bits set in the configured netmask value.
2. When an incoming IPv4 route address is evaluated against a route filter match criteria that uses the **address-mask** routing policy match type, the match succeeds if the following values are identical:
   - The bit-wise logical AND of the configured netmask value and the incoming IPv4 route address

- The bit-wise logical AND of the configured netmask value and the configured destination match prefix

For an example configuration of a route list that contains two **address-mask** match types, see "Example: Evaluation of an Address Mask Match Type with Longest-Match Lookup" on page 145.

How Prefix Order Affects Route List Evaluation

The order in which the prefixes are specified (from top to bottom) typically does not matter, because the policy framework software scans the route list looking for the longest prefix during evaluation. An exception to this rule is when you use the same destination prefix multiple times in a list. In this case, the order of the prefixes is important, because the list of identical prefixes is scanned from top to bottom, and the first match type that matches the route applies.

In the following example, different match types are specified for the same prefix. The route **0.0.0.0/0** would be rejected, the route **0.0.0.0/8** would be marked with **next-hop self**, and the route **0.0.0.0/25** would be rejected.

```
route-filter 0.0.0.0/0 upto /7 reject;
route-filter 0.0.0.0/0 upto /24 next-hop self;
route-filter 0.0.0.0/0 orlonger reject;
```

Common Configuration Problem with the Longest-Match Lookup

A common problem when defining a route list is including a shorter prefix that you want to match with a longer, similar prefix in the same list. For example, imagine that the prefix **192.168.254.0/24** is compared against the following route list:

```
route-filter 192.168.0.0/16 orlonger;
route-filter 192.168.254.0/23 exact;
```

Because the policy framework software performs longest-match lookup, the prefix **192.168.254.0/23** is determined to be the longest prefix. An exact match does not occur between **192.168.254.0/24** and **192.168.254.0/23 exact**. The software determines that the term does not match and goes on to the next term or routing policy, if present, or takes the accept or reject action specified by the default policy. (For more information about the default routing policies, see "Default Routing Policies and Actions" on page 16.) The shorter prefix **192.168.0.0/16 orlonger** that you wanted to match is inadvertently ignored.

One solution to this problem is to remove the prefix **192.168.0.0/16 orlonger** from the route list in this term and move it to another term where it is the only prefix or the longest prefix in the list.

Route List Examples

The examples in this section show only fragments of routing policies. Normally, you would combine these fragments with other terms or routing policies.

In all examples, remember that the following actions apply to nonmatching routes:

- Evaluate next term, if present.

- Evaluate next policy, if present.
- Take the accept or reject action specified by the default policy. For more information about the default routing policies, see "Default Routing Policies and Actions" on page 16.

The following examples show how to configure route lists for various purposes:

Example: Rejecting Routes with Specific Destination Prefixes and Mask Lengths

Reject routes with a destination prefix of **0.0.0.0** and a mask length from 0 through 8, and accept all other routes:

```
[edit]
policy-options {
   policy-statement policy-statement from-hall2 {
      term 1 {
         from {
            route-filter 0.0.0.0/0 upto /8 reject;
         }
      }
      then accept;
   }
}
```

Example: Rejecting Routes with a Mask Length Greater than Eight

Reject routes with a mask of **/8** and greater (that is, **/8**, **/9**, **/10**, and so on) that have the first 8 bits set to 0 and accept routes less than 8 bits in length:

```
[edit]
policy-options {
   policy-statement from-hall3 {
      term term1 {
```

```
        from {
          route-filter 0/0 upto /7 accept;
          route-filter 0/8 orlonger;
        }
        then reject;
      }
    }
  }
```

Example: Rejecting Routes with Mask Length Between 26 and 29

Reject routes with the destination prefix of **192.168.10/24** and a mask between **/26** and **/29** and accept all other routes:

```
[edit]
policy-options {
  policy-statement from-customer-a {
    term term1 {
      from {
        route-filter 192.168.10/24 prefix-length-range /26–/29 reject;
      }
      then accept;
    }
  }
}
```

Example: Rejecting Routes from Specific Hosts

Reject a range of routes from specific hosts, and accept all other routes:

```
[edit]
policy-options {
  policy-statement hosts-only {
    from {
      route-filter 10.125.0.0/16 upto /31 reject;
      route-filter 0/0;
    }
    then accept;
  }
}
```

You do not use the **through** match type in most routing policy configurations. You should think of **through** as a tool to group a contiguous set of exact matches. For example, instead of specifying four exact matches:

```
from route-filter 0.0.0.0/1 exact
from route-filter 0.0.0.0/2 exact
from route-filter 0.0.0.0/3 exact
from route-filter 0.0.0.0/4 exact
```

You could represent them with the following single match:

```
from route-filter 0.0.0.0/1 through 0.0.0.0/4
```

Example: Accepting Routes with a Defined Set of Prefixes

Explicitly accept a limited set of prefixes (in the first term) and reject all others (in the second term):

```
policy-options {
policy-statement internet-in {
   term 1 {
      from {
         route-filter 192.168.231.0/24 exact accept;
         route-filter 192.168.244.0/24 exact accept;
         route-filter 192.168.198.0/24 exact accept;
         route-filter 192.168.160.0/24 exact accept;
         route-filter 192.168.59.0/24 exact accept;
      }
   }
   term 2 {
      then {
         reject;
      }
   }
}
```

Example: Rejecting Routes with a Defined Set of Prefixes

Reject a few groups of prefixes, and accept the remaining prefixes:

```
[edit policy-options]
policy-statement drop-routes {
   term 1{
      from { # first, reject a number of prefixes:
         route-filter default exact reject; # reject 0.0.0.0/0 exact
         route-filter 0.0.0.0/8 orlonger reject; # reject prefix 0, mask /8 or longer
         route-filter 10.0.0.0/8 orlonger reject; # reject loopback addresses
         }
         route-filter 10.105.0.0/16 exact { # accept 10.105.0.0/16
            as-path-prepend “1 2 3”;
            accept;
         }
         route-filter 192.0.2.0/24 orlonger reject; # reject test network packets
         route-filter 224.0.0.0/3 orlonger reject; # reject multicast and higher
         route-filter 0.0.0.0/0 upto /24 accept; # accept everything up to /24
         route-filter 0.0.0.0/0 orlonger accept; # accept everything else
         }
      }
   }
}
```

Example: Rejecting Routes with Prefixes Longer than 24 Bits

Reject all prefixes longer than 24 bits. You would install this routing policy in a sequence of routing policies in an **export** statement. The first term in this filter passes on all routes with a prefix length of up to 24 bits. The second, unnamed term rejects everything else.

```
[edit policy-options]
policy-statement 24bit-filter {
```

```
    term acl20 {
      from {
        route-filter 0.0.0.0/0 upto /24;
      }
      then next policy;
    }
    then reject;
  }
```

If, in this example, you were to specify **route-filter 0.0.0.0/0 upto /24 accept**, matching prefixes would be accepted immediately and the next routing policy in the **export** statement would never get evaluated.

If you were to include the **then reject** statement in the term **acl20**, prefixes greater than 24 bits would never get rejected because the policy framework software, when evaluating the term, would move on to evaluating the next statement before reaching the **then reject** statement.

Example: Rejecting PIM Multicast Traffic Joins

Configure a routing policy for rejecting Protocol Independent Multicast (PIM) multicast traffic joins for a source destination prefix from a neighbor:

```
[edit]
policy-options {
  policy-statement join-filter {
    from {
      neighbor 10.14.12.20;
      source-address-filter 10.83.0.0/16 orlonger;
    }
    then reject;
  }
}
```

Example: Rejecting PIM Traffic

Configure a routing policy for rejecting PIM traffic for a source destination prefix from an interface:

```
[edit]
policy-options {
  policy-statement join-filter {
    from {
      interface so-1/0/0.0;
      source-address-filter 10.83.0.0/16 orlonger;
    }
    then reject;
  }
}
```

The following routing policy qualifiers apply to PIM:

- **interface**—Interface over which a join is received
- **neighbor**—Source from which a join originates

- **route-filter**—Group address
- **source-address-filter**—Source address for which to reject a join

For more information about importing a PIM join filter in a PIM protocol definition, see the *Junos OS Multicast Protocols Configuration Guide*.

Example: Accepting Incoming IPv4 Routes by Applying an Address Mask to the Route Address and the Destination Match Prefix

Accept incoming IPv4 routes with a destination prefix of **10.1.0/24** and the third byte an even number from 0 to 14, inclusive:

```
[edit]
policy-options {
  policy-statement from_customer_a {
    term term_1 {
      from {
        route-filter 10.1.0.0/24 address-mask 255.255.241.0;
      }
      then {
        ...
        reject;
      }
    }
  }
}
```

The route list in routing policy term **term_1** matches the following incoming IPv4 route addresses:

- **10.1.0.0/24**
- **10.1.2.0/24**
- **10.1.4.0/24**
- **10.1.6.0/24**
- **10.1.8.0/24**
- **10.1.10.0/24**
- **10.1.12.0/24**
- **10.1.14.0/24**

The bit-wise logical AND of the netmask value and the candidate route address must match the bit-wise logical AND of the netmask value and the match prefix address. That is, where the netmask bit pattern **255.255.241.0** contains a set bit, the incoming IPv4 route address being evaluated must match the value of the corresponding bit in the destination prefix address **10.1.0.0/24**.

- The first two bytes of the netmask value are binary **1111 1111 1111 1111**, which means that a candidate route address will fail the match if the first two bytes are not **10.1**.

- The third byte of the netmask value is binary **1111 0001**, which means that a candidate route address will fail the match if the third byte is greater than 15 (decimal), an odd number, or both.
- The prefix length of the match prefix address is **24** (decimal), which means that a candidate route address will fail the match if its prefix length is not exactly **24**.

As an example, suppose that the candidate route address being tested in the policy is **10.1.8.0/24** (binary **0000 1010 0000 0001 0000 1000**).

1. When the netmask value is applied to this candidate route address, the result is binary **0000 1010 0000 0001 0000 0000**.
2. When the netmask value is applied to the configured destination prefix address, the result is also binary **0000 1010 0000 0001 0000 0000**.
3. Because the results of both AND operations are the same, the match continues to the second match criteria.
4. Because the prefix lengths of the candidate address and the configured destination prefix address are the same (24 bits), the match succeeds.

As another example, suppose that the candidate route address being tested in the policy is **10.1.3.0/24** (binary **0000 1010 0000 0001 0000 0011**).

1. When the netmask value is applied to this candidate route address, the result is binary **0000 1010 0000 0001 0000 0001**.
2. However, when the netmask value is applied to the configured destination prefix address, the result is binary **0000 1010 0000 0001 0000 0000**.
3. Because the results of the two AND operations are different (in the third byte), the match fails.

Example: Accepting Incoming IPv4 Routes with Similar Patterns But Different Prefix Lengths

Accept incoming IPv4 route addresses of the form **10.*.1/24** or **10.*.1.*/32**:

```
[edit]
policy-options {
   policy-statement from_customer_b {
      term term_2 {
         from {
            route-filter 10.0.1.0/24 address-mask 255.0.255.0;
            route-filter 10.0.1.0/32 address-mask 255.0.255.0;
         }
         then {
            ...
            reject;
         }
      }
   }
}
```

The route filter match criteria **10.0.1.0/24 address-mask 255.0.255.0** matches an incoming IPv4 route address of the form **10.*.1/24**. The route's prefix length must be exactly 24 bits long, and any value is acceptable in the second byte.

The route filter match criteria **10.0.1.0/32 address-mask 255.0.255.0** matches an incoming IPv4 route address of the form **10.*.1.*/32**. The route's prefix length must be exactly 32 bits long, and any value is acceptable in the second byte and the fourth byte.

Example: Evaluation of an Address Mask Match Type with Longest-Match Lookup

This example illustrates how a longest-match lookup evaluates a route list that contains two **address-mask** match types. Consider the route list configured in the routing policy term **term_3** below:

```
[edit]
policy-options {
   policy-statement from_customer_c {
      term term_3 {
         from {
            route-filter 10.0.1.0/24 address-mask 255.0.255.0;
            route-filter 10.0.2.0/24 address-mask 255.240.255.0;
         }
         then {
            ...
         }
      }
   }
}
```

Suppose that the incoming IPv4 route source address **10.1.1.0/24** is tested against the route list configured in the policy term **term_3**:

1. The longest-match lookup tree for routing policy term **term_3** contains two match prefixes: one prefix for **10.0.1.0/24 address-mask 255.0.255.0** and one prefix for **10.0.2.0/24 address-mask 255.240.255.0**. When searching the tree for the longest-prefix match for a candidate, the longest-match lookup considers the number of contiguous high-order bits in the configured ***netmask-value*** instead of the length of the configured ***destination-prefix***:

   - For the first route filter match criteria, the longest-match lookup entry is **10.0.0.0/8** because the netmask value contains 8 contiguous high-order bits.
   - For second route filter match criteria, the longest-match lookup entry is **10.0.0.0/12** because the netmask value contains 12 contiguous high-order bits.

   For the candidate route address **10.1.1.0/24**, the longest-match lookup returns the tree entry **10.0.0.0/12**, which is corresponds to the route filter match criteria **10.0.2.0/24 address-mask 255.240.255.0**.

2. Now that the longest-match prefix in **term_3** has been identified for the candidate route address, the candidate route address is evaluated against the route filter match criteria **10.0.2.0/24 address-mask 255.240.255.0**:

   a. To test the incoming IPv4 route address **10.1.1.0/24**, the netmask value **255.240.255.0** is applied to **10.1.1.0/24**. The result is **10.0.1.0**.

b. To test the configured destination prefix address **10.0.2.0/24**, the netmask value **255.240.255.0** is applied to **10.0.2.0/24**. The result is **10.0.2.0**.

c. Because the results are different, the route filter match fails. No actions, whether specified with the match criteria or with the **then** statement, are taken. The incoming IPv4 route address is not evaluated against any other match criteria.

Configuring Subroutines in Routing Policy Match Conditions

You can use a routing policy called from another routing policy as a match condition. This process makes the called policy a *subroutine*.

For configuration instructions and examples, see the following section:

- Configuring Subroutines on page 146
- Example: Configuring a Subroutine on page 149

Configuring Subroutines

To configure a subroutine in a routing policy to be called from another routing policy, create the subroutine and specify its name using the **policy** match condition (described in "Configuring Match Conditions in Routing Policy Terms" on page 47) in the **from** or **to** statement of another routing policy:

```
[edit]
policy-options {
  policy-statement subroutine-policy-name {
    term term-name {
      from {
        match-conditions;
        prefix-list name;
        route-filter destination-prefix match-type <actions>;
        source-address-filter source-prefix match-type <actions>;
      }
      to {
        match-conditions;
      }
      then actions;
    }
  }
}
policy-options {
  policy-statement policy-name {
    term term-name {
      from {
        policy subroutine-policy-name;
      }
      to {
        policy subroutine-policy-name;
      }
      then actions;
    }
  }
```

```
}
```

NOTE: Do not evaluate a routing policy within itself. The result is that no prefixes ever match the routing policy.

The action specified in a subroutine is used to provide a match condition to the calling policy. If the subroutine specifies an action of accept, the calling policy considers the route to be a match. If the subroutine specifies an action of reject, the calling policy considers the route not to match. If the subroutine specifies an action that is meant to manipulate the route characteristics, the changes are made. For more details about the subroutine evaluation, see "How a Routing Policy Subroutine Is Evaluated" on page 24.

Possible Consequences of Termination Actions in Subroutines

A subroutine with particular statements can behave differently from a routing policy that contains the same statements. With a subroutine, you must remember that the possible termination actions of accept or reject specified by the subroutine or the default policy can greatly affect the expected results. (For more information about default routing policies, see "Default Routing Policies and Actions" on page 16.)

In particular, you must consider what happens if a match does not occur with routes specified in a subroutine and if the default policy action that is taken is the action that you expect and want.

For example, imagine that you are a network administrator at an Internet service provider (ISP) that provides service to Customer A. You have configured several routing policies for the different classes of neighbors that Customer A presents on various links. To save time maintaining the routing policies for Customer A, you have configured a subroutine that identifies their routes and various routing policies that call the subroutine, as shown below:

```
[edit]
policy-options {
  policy-statement customer-a-subroutine {
    from {
      route-filter 10.1/16 exact;
      route-filter 10.5/16 exact;
      route-filter 192.168.10/24 exact;
    }
    then accept;
  }
}
policy-options {
  policy-statement send-customer-a-default {
    from {
      policy customer-a-subroutine;
    }
    then {
      set metric 500;
      accept;
    }
  }
```

```
}
policy-options {
  policy-statement send-customer-a-primary {
    from {
      policy customer-a-subroutine;
    }
    then {
      set metric 100;
      accept;
    }
  }
}
policy-options {
  policy-statement send-customer-a-secondary {
    from {
      policy customer-a-subroutine;
    }
    then {
      set metric 200;
      accept;
    }
  }
}
protocols {
  bgp {
    group customer-a {
      export send-customer-a-default;
      neighbor 10.1.1.1;
      neighbor 10.1.2.1;
      neighbor 10.1.3.1 {
        export send-customer-a-primary;
      }
      neighbor 10.1.4.1 {
        export send-customer-a-secondary;
      }
    }
  }
}
```

The following results occur with this configuration:

- The group-level **export** statement resets the metric to **500** when advertising all BGP routes to neighbors **10.1.1.1** and **10.1.2.1** rather than just the routes that match the subroutine route filters.
- The neighbor-level **export** statements reset the metric to **100** and **200** when advertising all BGP routes to neighbors **10.1.3.1** and **10.1.4.1**, respectively, rather than just the BGP routes that match the subroutine route filters.

These unexpected results occur because the subroutine policy does not specify a termination action for routes that do not match the route filter and therefore, the default BGP export policy of accepting all BGP routes is taken.

If the statements included in this particular subroutine had been contained within the calling policies themselves, only the desired routes would have their metrics reset.

This example illustrates the differences between routing policies and subroutines and the importance of the termination action in a subroutine. Here, the default BGP export policy action for the subroutine was not carefully considered. A solution to this particular example is to add one more term to the subroutine that rejects all other routes that do not match the route filters:

```
[edit]
policy-options {
   policy-statement customer-a-subroutine {
      term accept-exact {
         from {
            route-filter 10.1/16 exact;
            route-filter 10.5/16 exact;
            route-filter 192.168.10/24 exact;
         }
         then accept;
      }
      term reject-others {
         then reject;
      }
   }
}
```

Termination action strategies for subroutines in general include the following:

- Depend upon the default policy action to handle all other routes.
- Add a term that accepts all other routes.
- Add a term that rejects all other routes.

The option that you choose depends upon what you want to achieve with your subroutine. Plan your subroutines carefully.

Example: Configuring a Subroutine

Create the subroutine **is-customer** and call it from the routing policies **export-customer** and **import-customer**. In **import-customer**, the action is taken only on routes that match the route filters defined in **is-customer**.

```
[edit]
policy-options {
   policy-statement is-customer {
      term match-customer {
         from {
            route-filter 10.100.1.0/24 exact;
            route-filter 10.186.100.0/24 exact;
         }
         then accept;
      }
      term drop-others {
         then reject;
      }
   }
   policy-statement export-customer {
      from policy is-customer;
```

```
    then accept;
  }
  policy-statement import-customer {
    from {
      protocol bgp;
      policy is-customer;
    }
    then {
      local-preference 10;
      accept;
    }
  }
}
```

Configuring Routing Policy Match Conditions Based on Routing Table Entries

In addition to defining match conditions in the **from** statement of a policy, you can use the **condition** statement to define conditions used during policy evaluation:

```
condition condition-name {
  if-route-exists address table table-name;
}
```

You can include this statement at the following hierarchy levels:

- **[edit policy-options]**
- **[edit logical-systems *logical-system-name* policy-options]**

To define a policy condition based on the existence of routes in specific tables for use in BGP export policies, specify a name for the condition and include the following options:

- **if-route-exists *address***—Specify the address of the route in question.
- **table *table-name***—Specify a routing table.

You can then add the defined condition to the **from** statement of a policy:

```
policy-options {
  policy-statement policy-name {
    term 1 {
      from {
        protocols bgp;
        condition condition-name;
      }
      then {
        accept;
      }
    }
    ...
  }
```

The **condition** statement is available on all platforms, but is limited to use in BGP export policies. To view the configured policy conditions and their associated routing tables and dependent routes, issue the **show policy conditions** operational mode command.

Related Documentation

- **show policy conditions** in the *Junos OS Routing Protocols and Policies Command Reference*

CHAPTER 10

Extended Actions Configuration

Prepending AS Numbers to BGP AS Paths

You can *prepend* one or more autonomous system (AS) numbers at the beginning of an AS path. The AS numbers are added at the beginning of the path after the actual AS number from which the route originates has been added to the path. Prepending an AS path makes a shorter AS path look longer and therefore less preferable to BGP.

In Junos OS Release 9.1 and later, you can specify 4-byte AS numbers as defined in RFC 4893, *BGP Support for Four-octet AS Number Space*, as well as the 2-byte AS numbers that are supported in earlier releases of the Junos OS. In plain-number format, you can configure a value in the range from 1 through 4,294,967,295.

If you have a router that does not support 4-byte AS numbers in the AS path, the prependend AS number displayed in the AS path is the AS_TRANS number, AS 23456. To display the route details, use the show route command. For more information, see Prepending 4-Byte AS Numbers in an AS Path in the *Using 4-Byte Autonomous System Numbers in BGP Networks Technology Overview* .

In the following example, from AS 1 there are two equal paths (through AS 2 and AS 3) to reach AS 4. You might want packets from certain sources to use the path through AS 2. Therefore, you must make the path through AS 3 look less preferable so that BGP chooses the path through AS 2. In the configuration for AS 1, prepend multiple AS numbers:

```
[edit]
policy-options {
  policy-statement as-path-prepend {
    term prepend {
      from {
        route-filter 192.168.0.0/16 orlonger;
        route-filter 172.16.0.0/12 orlonger;
        route-filter 10.0.0.0/8 orlonger;
      }
      then as-path-prepend "1 1 1 1";
    }
  }
}
```

Related Documentation

- *Junos OS Routing Protocols Configuration Guide*
- 4-Byte Autonomous System Numbers Overview in the *Using 4-Byte Autonomous System Numbers in BGP Networks Technology Overview*
- Prepending 4-Byte AS Numbers in an AS Path in the *Using 4-Byte Autonomous System Numbers in BGP Networks Technology Overview*

Adding AS Numbers to BGP AS Paths

You can expand or add one or more AS numbers to an AS sequence. The AS numbers are added before the local AS number has been added to the path. Expanding an AS path makes a shorter AS path look longer and therefore less preferable to BGP. The last AS number in the existing path is extracted and prepended *n* times, where *n* is a number from 1 through 32. This is similar to the AS path prepend action, except that the AS path expand action adds an arbitrary sequence of AS numbers.

For example, from AS 1 there are two equal paths (through AS 2 and AS 3) to reach AS 4. You might want packets from certain sources to use the path through AS 2. Therefore, you must make the path through AS 3 less preferable so that BGP chooses the path through AS 2. In AS 1, you can expand multiple AS numbers.

```
[edit]
policy-options {
  policy-statement as-path-expand {
    term expand {
      from {
        route-filter 192.168.0.0/16 orlonger;
        route-filter 172.16.0.0/12 orlonger;
        route-filter 10.0.0.0/8 orlonger;
      }
      then as-path-expand last-as count 4;
    }
  }
}
```

For routes from AS 2, this makes the route look like **1 2 2 2 2 2** when advertised, where **1** is from AS 1, the **2** from AS 2 is prepended four times, and the final **2** is the original **2** received from the neighbor router.

Using Routing Policies to Damp BGP Route Flapping

BGP *route flapping* describes the situation in which BGP systems send an excessive number of update messages to advertise network reachability information. BGP *flap damping* is a way to reduce the number of update messages sent between BGP peers, thereby reducing the load on these peers without adversely affecting the route convergence time.

Flap damping reduces the number of update messages by marking routes as ineligible for selection as the active or preferable route. Doing this leads to some delay, or *suppression*, in the propagation of route information, but the result is increased network stability. You typically apply flap damping to external BGP (EBGP) routes (that is, to routes in different ASs). You can also apply it within a confederation, between confederation member ASs. Because routing consistency within an AS is important, do not apply flap damping to IBGP routes. (If you do, it is ignored.)

BGP flap damping is defined in RFC 2439, *BGP Route Flap Damping*.

To effect changes to the default BGP flap damping values, you define actions by creating a named set of damping parameters and including it in a routing policy with the **damping** action (described in "Configuring Actions That Manipulate Route Characteristics" on page 56). For the damping routing policy to work, you also must enable BGP route flap damping.

The following sections discuss the following topics:

Configuring BGP Flap Damping Parameters

To define damping parameters, include the **damping** statement:

```
damping name {
    disable;
    half-life minutes;
    max-suppress minutes;
    reuse number;
    suppress number;
}
```

You can include this statement at the following hierarchy levels:

- **[edit policy-options]**
- **[edit logical-systems *logical-system-name* policy-options]**

The name identifies the group of damping parameters. It can contain letters, numbers, and hyphens (-) and can be up to 255 characters. To include spaces in the name, enclose the entire name in quotation marks (" ").

You can specify one or more of the damping parameters described in Table 18 on page 156.

Table 18: Damping Parameters

Damping Parameter	Description	Default	Possible Values
half-life ***minutes***	Decay half-life, in minutes	15 minutes	1 through 45 minutes
max-suppress ***minutes***	Maximum hold-down time, in minutes	60 minutes	1 through 720 minutes
reuse	Reuse threshold	750 (unitless)	1 through 20,000 (unitless)
suppress	Cutoff (suppression) threshold	3000 (unitless)	1 through 20,000 (unitless)

If you do not specify one or more of the damping parameters, the default value of the parameter is used.

To understand how to configure these parameters, you need to understand how damping suppresses routes. How long a route can be suppressed is based on a *figure of merit*, which is a value that correlates to the probability of future instability of a route. Routes with higher figure-of-merit values are suppressed for longer periods of time. The figure-of-merit value decays exponentially over time.

A figure-of-merit value of zero is assigned to each new route. The value is increased each time the route is withdrawn or readvertised, or when one of its path attributes changes. With each incident of instability, the value increases as follows:

- Route is withdrawn—1000
- Route is readvertised—1000
- Route's path attributes change—500

NOTE: Other vendors' implementations for figure-of-merit increase the value only when a route is withdrawn. The Junos OS implementation for figure-of-merit increases the value for both route withdrawal and route readvertisement. To accommodate other implementations for figure-of-merit, multiply the reuse and suppress threshold values by 2.

When a route's figure-of-merit value reaches a particular level, called the *cutoff* or *suppression threshold*, the route is suppressed. If a route is suppressed, the routing table no longer installs the route into the forwarding table and no longer exports this route to any of the routing protocols. By default, a route is suppressed when its figure-of-merit value reaches 3000. To modify this default, include the **suppress** option at the **[edit policy-options damping** ***name*****]** hierarchy level.

If a route has flapped, but then becomes stable so that none of the incidents listed previously occur within a configurable amount of time, the figure-of-merit value for the

route decays exponentially. The default half-life is 15 minutes. For example, for a route with a figure-of-merit value of 1500, if no incidents occur, its figure-of-merit value is reduced to 750 after 15 minutes and to 375 after another 15 minutes. To modify the default half-life, include the **half-life** option at the **[edit policy-options damping *name*]** hierarchy level.

NOTE: For the half-life, configure a value that is less than the max-suppress. If you do not, the configuration is rejected.

A suppressed route becomes reusable when its figure-of-merit value decays to a value below a *reuse threshold*, thus allowing routes that experience transient instability to once again be considered valid. The default reuse threshold is 750. When the figure-of-merit value passes below the reuse threshold, the route once again is considered usable and can be installed in the forwarding table and exported from the routing table. To modify the default reuse threshold, include the **reuse** option at the **[edit policy-options damping *name*]** hierarchy level.

The maximum suppression time provides an upper bound on the time that a route can remain suppressed. The default maximum suppression time is 60 minutes. To modify the default, include the **max-suppress** option at the **[edit policy-options damping *name*]** hierarchy level.

NOTE: For the max-suppress, configure a value that is greater than the half-life. If you do not, the configuration is rejected.

A route's figure-of-merit value stops increasing when it reaches a maximum suppression threshold, which is determined based on the route's suppression threshold level, half-life, reuse threshold, and maximum hold-down time.

The merit ceiling, ε_c, which is the maximum merit that a flapping route can collect, is calculated using the following formula:

$$\varepsilon_c \leq \varepsilon_r e^{(t/\lambda)(\ln 2)}$$

ε_r is the figure-of-merit reuse threshold, **t** is the maximum hold-down time in minutes, and λ is the half-life in minutes. For example, if you use the default figure-of-merit values in this formula, but use a half-life of 30 minutes, the calculation is as follows:

$$\varepsilon_c \leq 750\, e^{(60/30)(\ln 2)}$$

$$\varepsilon_c \leq 3000$$

NOTE: The cutoff threshold, which you configure using the suppress option, must be less than or equal to the merit ceiling, ε_c. If the configured cutoff threshold or the default cutoff threshold is greater than the merit ceiling, the route is never suppressed and damping never occurs.

To display figure-of-merit information, use the **show policy damping** command.

A route that has been assigned a figure of merit is considered to have a damping state. To display the current damping information on the routing device, use the **show route detail** command.

Specifying BGP Flap Damping as the Action in Routing Policy Terms

To BGP flap damping as the action in a routing policy term, include the **damping** statement and the name of the configured damping parameters either as an option of the **route-filter** statement at the **[edit policy-options policy-statement** *policy-name* **term** *term-name* **from]** hierarchy level:

```
[edit policy-options policy-statement policy-name term term-name from]
route-filter destination-prefix match-type {
  damping damping-parameters;
}
```

or at the **[edit policy-options policy-statement** *policy-name* **term** *term-name* **then]** hierarchy level:

```
[edit policy-options policy-statement policy-name term term-name then]
damping damping-parameters;
```

Disabling Damping for Specific Address Prefixes

Normally, you enable or disable damping on a per-peer basis. However, you can disable damping for a specific prefix received from a peer by including the **disable** option:

```
disable;
```

You can include this statement at the following hierarchy levels:

- **[edit policy-options damping** *name***]**
- **[edit logical-systems** *logical-system-name* **policy-options damping** *name***]**

Example: Disabling Damping for a Specific Address Prefix

In this routing policy example, although damping is enabled for the peer, the **damping none** statement specifies that damping be disabled for prefix **10.0.0.0/8** in **Policy-A**. This route is not damped because the routing policy statement named **Policy-A** filters on the prefix **10.0.0.0/8** and the action points to the **damping** statement named **none**. The remaining prefixes are damped using the default parameters.

```
[edit]
policy-options {
  policy-statement Policy-A {
    from {
      route-filter 10.0.0.0/8 exact;
    }
    then damping none;
  }
  damping none {
    disable;
  }
```

```
}
```

Example: Configuring BGP Flap Damping

Enable BGP flap damping and configure damping parameters:

```
[edit]
routing-options {
  autonomous-system 666;
}
protocols {
  bgp {
    damping;
    group group1 {
      traceoptions {
        file bgp-log size 1m files 10;
        flag damping;
      }
      import damp;
      type external;
      peer-as 10458;
      neighbor 192.168.2.30;
    }
  }
}
policy-options {
  policy-statement damp {
    from {
      route-filter 192.168.0.0/32 exact {
        damping high;
        accept;
      }
      route-filter 172.16.0.0/32 exact {
        damping medium;
        accept;
      }
      route-filter 10.0.0.0/8 exact {
        damping none;
        accept;
      }
    }
  }
  damping high {
    half-life 30;
    suppress 3000;
    reuse 750;
    max-suppress 60;
  }
  damping medium {
    half-life 15;
    suppress 3000;
    reuse 750;
    max-suppress 45;
  }
  damping none {
    disable;
```

```
    }
  }
```

To display damping parameters for this configuration, use the **show policy damping** command:

```
user@host> show policy damping
Damping information for "high":
  Halflife: 30 minutes
  Reuse merit: 750 Suppress/cutoff merit: 3000
  Maximum suppress time: 60 minutes
  Computed values:
    Merit ceiling: 3008
    Maximum decay: 24933
Damping information for "medium":
  Halflife: 15 minutes
  Reuse merit: 750 Suppress/cutoff merit: 3000
  Maximum suppress time: 45 minutes
  Computed values:
    Merit ceiling: 6024
    Maximum decay: 12449
Damping information for "none":
Damping disabled
```

Configuring Per-Packet Load Balancing

To configure per-packet load balancing as described in "Overview of Per-Packet Load Balancing" on page 33, include the **load-balance per-packet** statement either as an option of the **route-filter** statement at the **[edit policy-options policy-statement *policy-name* term *term-name* from]** hierarchy level:

```
[edit policy-options policy-statement policy-name term term-name from]
route-filter destination-prefix match-type {
  load-balance per-packet;
}
```

or at the **[edit policy-options policy-statement *policy-name* term *term-name* then]** hierarchy level:

```
[edit policy-options policy-statement policy-name term term-name then]
load-balance per-packet;
```

To complete the configuration you must apply the routing policy to routes exported from the routing table to the forwarding table, by including the policy name in the list specified by the **export** statement:

```
export [ policy-names ];
```

You can include this statement at the following hierarchy levels:

- **[edit routing-options forwarding-table]**
- **[edit routing-instances *routing-instance-name* routing-options forwarding-table]**
- **[edit logical-systems *logical-system-name* routing-options forwarding-table]**
- **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* routing-options forwarding-table]**

By default, the software ignores port data when determining flows. To enable per-flow load balancing, you must set the **load-balance per-packet** action in the routing policy configuration.

To include port data in the flow determination, include the **family inet** statement at the **[edit forwarding-options hash-key]** hierarchy level:

```
[edit forwarding-options hash-key]
family inet {
   layer-3;
   layer-4;
}
```

If you include both the **layer-3** and **layer-4** statements, the router uses the following Layer 3 and Layer 4 information to load-balance:

- Source IP address
- Destination IP address
- Protocol
- Source port number
- Destination port number
- Incoming interface index
- IP type of service

The router recognizes packets in which all of these **layer-3** and **layer-4** parameters are identical, and ensures that these packets are sent out through the same interface. This prevents problems that might otherwise occur with packets arriving at their destination out of their original sequence.

This is appropriate behavior for Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) packets. For Internet Control Message Protocol (ICMP) packets, the field location offset is the checksum field, which makes each ping packet a separate "flow." There are other protocols that can be encapsulated in IP that may have a varying value in the 32-bit offset. This may also be problematic because these protocols are seen as a separate flow.

With M Series (with the exception of the M120 router) and T Series routers, the first fragment is mapped to the same load-balanced destination as the unfragmented packets. The other fragments can be mapped to other load-balanced destinations.

For the M120 router only, all fragments are mapped to the same load-balanced destination. This destination is not necessarily the same as that for unfragmented packets.

By default, or if you include only the **layer-3** statement, the router uses the incoming interface index as well as the following Layer 3 information in the packet header to load balance traffic:

- Source IP address
- Destination IP address

- Protocol

By default, IP version 6 (IPv6) packets are automatically load-balanced based on the following Layer 3 and Layer 4 information:

- Source IP address
- Destination IP address
- Protocol
- Source port number
- Destination port number
- Incoming interface index
- Traffic class

Per-Packet Load Balancing Examples

Perform per-packet load balancing for all routes:

```
[edit]
policy-options {
  policy-statement load-balancing-policy {
    then {
      load-balance per-packet;
    }
  }
}
routing-options {
  forwarding-table {
    export load-balancing-policy;
  }
}
```

Perform per-packet load balancing only for a limited set of routes:

```
[edit]
policy-options {
  policy-statement load-balancing-policy {
    from {
      route-filter 192.168.10/24 orlonger;
      route-filter 10.114/16 orlonger;
    }
    then {
      load-balance per-packet;
    }
  }
}
routing-options {
  forwarding-table {
    export load-balancing-policy;
  }
}
```

Configuring Load Balancing Based on MPLS Labels

Juniper Networks routers can load-balance on a per-packet basis in MPLS. Load balancing can be performed on information in both the IP header and on up to three MPLS labels, providing a more uniform distribution of MPLS traffic to next hops. This feature is enabled on supported platforms by default and requires no configuration.

Load balancing is used to evenly distribute traffic when the following conditions apply:

- There are multiple equal-cost next hops over different interfaces to the same destination.
- There is a single next hop over an aggregated interface.

By default, when load balancing is used to help distribute traffic, Junos OS employs a hash algorithm to select a next-hop address to install into the forwarding table. Whenever the set of next hops for a destination changes in any way, the next-hop address is reselected by means of the hash algorithm. You can configure how the hash algorithm is used to load-balance traffic across a set of equal-cost label switched paths (LSPs).

An LSP tends to load-balance its placement by randomly selecting one of the equal-cost next hops and using it exclusively. The random selection is made independently at each transit router, which compares Interior Gateway Protocol (IGP) metrics alone. No consideration is given to bandwidth or congestion levels.

To load-balance based on the MPLS label information, configure the **family mpls** statement:

```
[edit forwarding-options hash-key]
family mpls {
    label-1;
    label-2;
    label-3;
    no-labels;
    no-label-1-exp;
    payload {
        ether-pseudowire;
        ip {
            layer-3-only;
            port-data {
                destination-lsb;
                destination-msb;
                source-lsb;
                source-msb;
            }
        }
    }
}
```

You can include this statement at the following hierarchy levels:

- **[edit logical-systems *logical-system-name* forwarding-options hash-key]**
- **[edit forwarding-options hash-key]**

This feature applies to aggregated Ethernet and aggregated SONET/SDH interfaces as well as multiple equal-cost MPLS next hops. In addition, on the T Series, MX Series, M120, and M320 routers only, you can configure load balancing for IPv4 traffic over Layer 2 Ethernet pseudowires. You can also configure load balancing for Ethernet pseudowires based on IP information. The option to include IP information in the hash key provides support for Ethernet circuit cross-connect (CCC) connections.

Table 19 on page 164 provides detailed information about all of the possible MPLS LSP load-balancing options.

Table 19: MPLS LSP Load Balancing Options

Statement	MPLS LSP Load Balancing Options
label-l	Include the first label in the hash key. Use this option for single label packets.
label-2	Include the second label in the hash key. You must also configure the **label-1** option. The entire first label and the first 16 bits of the second label are used in the hash key.
label-3	Include the third label in the hash key. You must also configure the **label-1** option and the **label-2** option.
no-labels	Excludes MPLS labels from the hash key.
no-label-1-exp	Excludes the EXP bit of the top label from the hash key. You must also configure the **label-l** option. For Layer 2 VPNs, the router could encounter a packet reordering problem. When a burst of traffic pushes the customer traffic bandwidth to exceed its limits, the traffic might be affected in mid flow. Packets might be reordered as a result. By excluding the EXP bit from the hash key, you can avoid this reordering problem.
payload	Allows you to configure which parts of the IP packet payload to include in the hash key.
ether-pseudowire	(M120, M320, MX Series, and T Series routers only5)—Load-balance IPv4 traffic over Layer 2 Ethernet pseudowires.
ip	Include the IPv4 or IPv6 address in the hash key. You must also configure either **label-l** or **no-labels**.
layer-3-only	Include only the Layer 3 IP information in the hash key. Excludes all of the **port-data** bytes from the hash key.
port-data	Include the source and destination port field information. By default, the most significant byte and least significant byte of the source and destination port fields are used in the hash key. To select specific bytes to use in the hash key, include one or more of the **source-msb**, **source-lsb**, **destination-msb**, and **destination-lsb** options at the **[edit forwarding-options hash-key family mpls payload ip port-data]** hierarchy level. To prevent all four bytes from being hashed, include the **layer-3-only** statement at the **[edit forwarding-options hash-key family mpls payload ip]** hierarchy level.
destination-lsb	Include the least significant byte of the destination port in the hash key. Can be combined with any of the other **port-data** options.
destination-msb	Include the most significant byte of the destination port in the hash key. Can be combined with any of the other **port-data** options.

Table 19: MPLS LSP Load Balancing Options *(continued)*

Statement	MPLS LSP Load Balancing Options
source-lsb	Include the least significant byte of the source port in the hash key. Can be combined with any of the other **port-data** options.
source-msb	Include the most significant byte of the source port in the hash key. Can be combined with any of the other **port-data** options.

The following examples illustrate ways in which you can configure MPLS LSP load balancing:

- To include the IP address as well as the first label in the hash key, configure the **label-1** statement and the **ip** option for the **payload** statement at the **[edit forwarding-options hash-key family mpls]** hierarchy level:

```
[edit forwarding-options hash-key family mpls]
label-1;
payload {
   ip;
}
```

- To include the IP address as well as both the first and second labels in the hash key, configure the **label-1** and **label-2** options and the **ip** option for the **payload** statement at the **[edit forwarding-options hash-key family mpls]** hierarchy level:

```
[edit forwarding-options hash-key family mpls]
label-1;
label-2;
payload {
   ip;
}
```

NOTE: You can include this combination of statements on M320 and T Series Core Routers only. If you include them on an M Series Multiservice Edge Router, only the first MPLS label and the IP payload are used in the hash key.

- For Layer 2 VPNs, the router could encounter a packet reordering problem. When a burst of traffic pushes the customer traffic bandwidth to exceed its limits, the traffic might be affected in mid flow. Packets might be reordered as a result. By excluding the EXP bit from the hash key, you can avoid this reordering problem. To exclude the EXP bit of the first label from the hash calculations, include the **no-label-1-exp** statement at the **[edit forwarding-options hash-key family mpls]** hierarchy level:

```
[edit forwarding-options hash-key family mpls]
label-1;
no-label-1-exp;
payload {
   ip;
}
```

Related Documentation

- Configuring Load Balancing for Ethernet Pseudowires on page 166

Configuring Load Balancing for Ethernet Pseudowires

You can configure load balancing for IPv4 traffic over Layer 2 Ethernet pseudowires. You can also configure load balancing for Ethernet pseudowires based on IP information. The option to include IP information in the hash key provides support for Ethernet circuit cross-connect (CCC) connections.

NOTE: This feature is supported only on M120, M320, MX Series, and T Series routers.

To configure load balancing for IPv4 traffic over Layer 2 Ethernet pseudowires, include the **ether-pseudowire** statement at the **[edit forwarding-options hash-key family mpls payload]** hierarchy level:

```
[edit forwarding-options]
hash-key {
  family mpls {
    (label-1 | no-labels);
    payload {
      ether-pseudowire;
    }
  }
}
```

NOTE: You must also configure either the **label-1** or the **no-labels** statement at the **[edit forwarding-options hash-key family mpls]** hierarchy level.

You can also configure load balancing for Ethernet pseudowires based on IP information. This functionality provides support for load balancing for Ethernet cross-circuit connect (CCC) connections. To include IP information in the hash key, include the **ip** statement at the **[edit forwarding-options hash-key family mpls payload]** hierarchy level:

```
[edit forwarding-options]
hash-key {
  family mpls {
    (label-1 | no-labels);
    payload {
      ip;
    }
  }
}
```

NOTE: You must also configure either the **label-1** or **no-labels** statement at the **[edit forwarding-options hash-key family mpls]** hierarchy level.

You can configure load balancing for IPv4 traffic over Ethernet pseudowires to include only Layer 3 IP information in the hash key. To include only Layer 3 IP information, include the **layer-3-only** option at the **[edit forwarding-options family mpls hash-key payload ip]** hierarchy level:

```
[edit forwarding-options]
hash-key {
  family mpls {
    (label-1 | no-labels);
    payload {
      ip {
        layer-3-only;
      }
    }
  }
}
```

NOTE: You must also configure either the **label-1** or **no-labels** statement at the **[edit forwarding-options hash-key family mpls]** hierarchy level.

Configuring Load Balancing Based on MAC Addresses

The hash key mechanism for load-balancing uses Layer 2 media access control (MAC) information such as frame source and destination address. To load-balance traffic based on Layer 2 MAC information, include the **family multiservice** statement at the **[edit forwarding-options hash-key]** hierarchy level:

```
family multiservice {
  destination-mac;
  source-mac;
}
```

To include the destination-address MAC information in the hash key, include the **destination-mac** option. To include the source-address MAC information in the hash key, include the **source-mac** option.

NOTE: Any packets that have the same source and destination address will be sent over the same path.

NOTE: You can configure per-packet load balancing to optimize VPLS traffic flows across multiple paths.

NOTE: J Series Services Routers do not support this feature.

Related Documentation

- *Junos OS VPNs Configuration Guide*

Configuring VPLS Load Balancing Based on IP and MPLS Information

In Junos OS Release 9.4 and later, you can configure load balancing for VPLS traffic to have the hash key include IP information and MPLS labels on the M120 and M320 routers only. In earlier Junos OS Releases, you can configure load balancing based only on Layer 2 information. In Junos OS Release 9.5 and later, you can configure load balancing for VPLS traffic based on Layer 3 IP and Layer 4 information on MX Series routers only. For more information, see "Configuring VPLS Load Balancing on MX Series 3D Universal Edge Routers" on page 169.

For IPv4 traffic, only the IP source and destination addresses are included in the hash key. For MPLS and IPv4 traffic, one or two MPLS labels and IPv4 source and destination addresses are included. For MPLS Ethernet pseudowires, only one or two MPLS labels are included in the hash key.

NOTE: VPLS load balancing based on MPLS labels and IP information is supported only on the M120 and M320 routers. In Junos OS Release 9.5 and later, on MX Series routers only, you can configure VPLS load balancing based on IP and Layer 4 information.

To optimize VPLS flows across multiple paths based on IP and MPLS information, include the **family multiservice** statement at the **[edit forwarding-options hash-key]** hierarchy level:

```
[edit forwarding-options hash-key]
family multiservice {
    label-1;
    label-2;
    payload {
        ip {
            layer-3-only;
        }
    }
}
```

To use the first MPLS label in the hash key, include the **label-1** statement:

```
[edit forwarding-options hash-key family multiservice]
label-1;
```

To use the second MPLS label, include both the **label-1** and **label-2** statements:

```
[edit forwarding-options hash-key family multiservice]
label-1;
label-2;
```

To use the packet's IPv4 payload in the hash key, include the **payload** and **ip** statements:

```
[edit forwarding-options hash-key family multiservice]
```

```
payload {
  ip;
}
```

NOTE: Only IPv4 is supported.

To include only Layer 3 information from the IPv4 payload, specify the **layer-3-only** option to the **payload ip** statement:

```
[edit forwarding-options hash-key family multiservice]
payload {
  ip {
    layer-3-only;
  }
}
```

To use the first and second MPLS labels and the packet's IP payload in the hash key, include the **label-1**, **label-2**, and **payload ip** statements:

```
[edit forwarding-options hash-key family multiservice]
label-1;
label-2;
payload {
  ip;
}
```

Configuring VPLS Load Balancing on MX Series 3D Universal Edge Routers

In Junos OS Release 9.5 and later, on MX Series routers, you can configure the load balancing hash key for Layer 2 traffic to use fields in the Layer 3 and Layer 4 headers inside the frame payload. You can also configure VPLS load balancing based on IP and MPLS information on M120 and M320 routers only. For more information, see "Configuring VPLS Load Balancing Based on IP and MPLS Information" on page 168.

You can configure load balancing on MX Series routers based on Layer 3 or Layer 4 information or both.

To configure VPLS load balancing on the MX Series router to include either Layer 3 IP information or Layer 4 headers or both:

1. Include the **payload** statement at the **[edit forwarding-options hash-key family multiservice]** hierarchy level.
2. Include the **ip** statement at the **[edit forwarding-options hash-key family multiservice payload]** hierarchy level.

To configure VPLS load balancing to include the Layer 3 information:

1. Include the **layer-3** statement at the **[edit forwarding-options hash-key family multiservice payload ip]** hierarchy level.

2. Include the **source-address-only** statement at the **[edit forwarding-options hash-key family multiservice payload ip layer-3]** hierarchy level to include information about the IP source address only in the hash key.
3. Include **destination-address-only** statement at the **[edit forwarding-options hash-key family multiservice payload ip layer-3]** hierarchy level to include information about the IP destination address only in the hash key.

NOTE: **You can configure either the source-address-only or the destination-address-only statements at a time, not both. They are mutually exclusive.**

To configure VPLS load balancing to include Layer 4 information:

- Include the **layer-4** statement at the **[edit forwarding-options hash-key family multiservice payload ip]** hierarchy level.

The following example shows load balancing configured to use the source Layer 3 IP address option and Layer 4 header fields as well as the source and destination MAC addresses:

```
[edit forwarding-options hash-key]
family multiservice {
   source-mac;
   destination-mac;
   payload {
      ip {
         layer-3 {
            source-address-only;
         }
         layer-4;
      }
   }
}
```

Related Documentation

- family multiservice on page 270
- hash-key on page 281

Example: Configuring Multicast Load Balancing over Aggregated Ethernet Links

This example shows how to configure point-to-multipoint (P2MP) LSPs to load-balance across aggregated Ethernet links. The load balancing applies to all traffic types, including multicast. Feature parity for multicast load balancing of point-to-multipoint LSPs over aggregated Ethernet child links on the Junos Trio chipset is supported in Junos OS Releases 11.1R2, 11.2R2, and 11.4.

- Requirements on page 171
- Overview on page 171

Requirements

Before you begin:

- Configure the router interfaces. See the *Junos OS Network Interfaces Configuration Guide*.
- Configure an interior gateway protocol or static routing. See the *Junos OS Routing Protocols Configuration Guide*.

Overview

This example shows a sample topology and configuration to perform the following tasks:

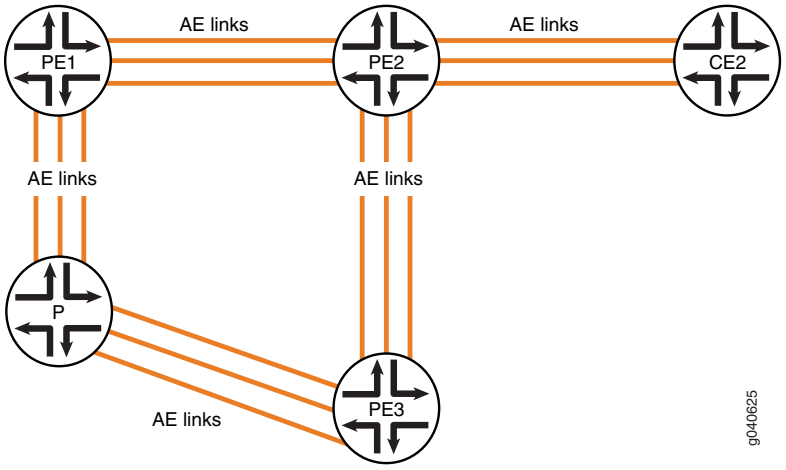
- Load balancing VPLS multicast traffic over link aggregation
- Load balancing P2MP multicast traffic over link aggregation
- Re-load balancing after a change in the nexthop topology

  Nexthop topology changes might include but are not limited to:

  - L2 membership change in the link aggregation
  - Indirect nexthop change
  - Composite nexthop change

Figure 14 on page 171 shows the topology for this example. The example includes the configuration for PE1 and PE2.

Figure 14: Multicast Load Balancing over Aggregated Ethernet Links



Configuration

CLI Quick Configuration

To quickly configure load balancing across aggregated Ethernet links, copy the following commands into a text file, remove any line breaks, and then paste the commands into the CLI.

PE1:

```
[edit]
set forwarding-options hash-key family multiservice source-mac
set forwarding-options hash-key family multiservice destination-mac
set forwarding-options hash-key family multiservice payload ip layer-3
set interfaces ge-0/0/6 gigether-options 802.3ad ae0
set interfaces ge-0/1/6 gigether-options 802.3ad ae0
set interfaces ge-0/2/2 encapsulation ethernet-vpls
set interfaces ge-0/2/2 unit 0 family vpls
set interfaces ge-0/2/3 gigether-options 802.3ad ae0
set interfaces ge-0/2/6 gigether-options 802.3ad ae0
set interfaces ge-0/3/0 gigether-options 802.3ad ae0
set interfaces ge-0/3/1 gigether-options 802.3ad ae0
set interfaces ge-0/3/6 gigether-options 802.3ad ae0
set interfaces ge-1/0/6 gigether-options 802.3ad ae0
set interfaces ge-1/2/6 unit 0 family inet address 13.1.1.2/30
set interfaces ae0 unit 0 family inet address 11.11.11.1/30
set interfaces ae0 unit 0 family iso
set interfaces ae0 unit 0 family mpls
set policy-options policy-statement exp-to-fwd term a from community grn-com
set policy-options policy-statement exp-to-fwd term a then install-nexthop lsp PE1-to-PE2
set policy-options policy-statement exp-to-fwd term a then accept
set policy-options community grn-com members target:65000:1
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable
set protocols mpls label-switched-path PE1-to-PE2 to 10.255.19.77
set protocols mpls label-switched-path PE1-to-PE3 to 10.255.19.79
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp group int type internal
set protocols bgp group int local-address 10.255.71.214
set protocols bgp group int family inet any
set protocols bgp group int family l2vpn signaling
set protocols bgp group int neighbor 10.255.19.77
set protocols bgp group int neighbor 10.255.19.79
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set routing-instances vpls instance-type vpls
set routing-instances vpls interface ge-0/2/2.0
set routing-instances vpls route-distinguisher 65000:1
set routing-instances vpls vrf-target target:65000:1
set routing-instances vpls protocols vpls site-range 3
set routing-instances vpls protocols vpls no-tunnel-services
set routing-instances vpls protocols vpls site asia site-identifier 1
set routing-instances vpls protocols vpls site asia interface ge-0/2/2.0
set routing-instances vpls protocols vpls vpls-id 100
```

PE2:

```
[edit]
set interfaces ge-0/0/7 gigether-options 802.3ad ae0
set interfaces ge-0/1/7 gigether-options 802.3ad ae0
set interfaces ge-0/2/3 gigether-options 802.3ad ae0
set interfaces ge-0/2/7 gigether-options 802.3ad ae0
```

```
set interfaces ge-2/0/0 gigether-options 802.3ad ae1
set interfaces ge-2/0/1 gigether-options 802.3ad ae1
set interfaces ge-2/0/2 gigether-options 802.3ad ae1
set interfaces ge-2/0/4 encapsulation ethernet-vpls
set interfaces ge-2/0/4 unit 0 family vpls
set interfaces ge-2/0/7 gigether-options 802.3ad ae0
set interfaces ge-2/0/9 unit 0 family inet address 1.1.1.1/30
set interfaces ge-2/0/9 unit 0 family mpls
set interfaces ge-2/1/7 gigether-options 802.3ad ae0
set interfaces ge-2/2/7 gigether-options 802.3ad ae0
set interfaces ge-2/3/7 gigether-options 802.3ad ae0
set interfaces ae0 unit 0 family inet address 11.11.11.2/30
set interfaces ae0 unit 0 family iso
set interfaces ae0 unit 0 family mpls
set interfaces ae1 unit 0 family inet address 10.1.1.1/30
set interfaces ae1 unit 0 family mpls
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable
set protocols mpls label-switched-path PE2-to-PE3 from 10.255.19.77
set protocols mpls label-switched-path PE2-to-PE3 to 10.255.19.79
set protocols mpls label-switched-path PE2-to-PE1 to 10.255.71.214
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp group int type internal
set protocols bgp group int local-address 10.255.19.77
set protocols bgp group int family inet any
set protocols bgp group int family l2vpn signaling
set protocols bgp group int neighbor 10.255.71.214
set protocols bgp group int neighbor 10.255.19.79
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface ge-2/0/0.0
set protocols ospf area 0.0.0.0 interface ge-2/0/1.0
set protocols ospf area 0.0.0.0 interface ge-2/0/2.0
set protocols ospf area 0.0.0.0 interface ae0.0
set protocols ospf area 0.0.0.0 interface ae1.0
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set routing-instances vpls instance-type vpls
set routing-instances vpls interface ge-2/0/4.0
set routing-instances vpls route-distinguisher 65000:1
set routing-instances vpls vrf-target target:65000:1
set routing-instances vpls protocols vpls site-range 3
set routing-instances vpls protocols vpls no-tunnel-services
set routing-instances vpls protocols vpls site 2 site-identifier 2
set routing-instances vpls protocols vpls site 2 interface ge-2/0/4.0
set routing-instances vpls protocols vpls vpls-id 100
```

Step-by-Step Procedure

To configure PE1:

1. Configure PE1's interfaces.

```
[edit]
user@host# edit interfaces
```

```
[edit interfaces]
user@host# set ge-0/0/6 gigether-options 802.3ad ae0
user@host# set ge-0/1/6 gigether-options 802.3ad ae0
user@host# set ge-0/2/2 encapsulation ethernet-vpls
user@host# set ge-0/2/2 unit 0 family vpls
user@host# set ge-0/2/3 gigether-options 802.3ad ae0
user@host# set ge-0/2/6 gigether-options 802.3ad ae0
user@host# set ge-0/3/0 gigether-options 802.3ad ae0
user@host# set ge-0/3/1 gigether-options 802.3ad ae0
user@host# set ge-0/3/6 gigether-options 802.3ad ae0
user@host# set ge-1/0/6 gigether-options 802.3ad ae0
user@host# set ge-1/2/6 unit 0 family inet address 13.1.1.2/30
user@host# set ae0 unit 0 family inet address 11.11.11.1/30
user@host# set ae0 unit 0 family iso
user@host# set ae0 unit 0 family mpls
user@host# exit
```

2. On PE1, configure the packet header data to be used for per-flow load balancing.

```
[edit]
user@host# edit forwarding-options hash-key family multiservice
[edit forwarding-options hash-key family multiservice]
set source-mac
set destination-mac
set payload ip layer-3
user@host# exit
```

3. Configure the routing policy on PE1.

```
[edit]
user@host# edit policy-options
[edit policy-options]
user@host# set policy-statement exp-to-fwd term a from community grn-com
user@host# set policy-statement exp-to-fwd term a then install-nexthop lsp
  PE1-to-PE2
user@host# set policy-statement exp-to-fwd term a then accept
user@host# set policy-options community grn-com members target:65000:1
user@host# exit
```

4. Configure PE1's routing protocols and MPLS.

```
[edit]
user@host# edit protocols
[edit protocols]
user@host# set rsvp interface all
user@host# set rsvp interface fxp0.0 disable
user@host# set mpls label-switched-path PE1-to-PE2 to 10.255.19.77
user@host# set mpls label-switched-path PE1-to-PE3 to 10.255.19.79
user@host# set mpls interface all
user@host# set mpls interface fxp0.0 disable
user@host# set bgp group int type internal
user@host# set bgp group int local-address 10.255.71.214
user@host# set bgp group int family inet any
user@host# set bgp group int family l2vpn signaling
user@host# set bgp group int neighbor 10.255.19.77
user@host# set bgp group int neighbor 10.255.19.79
user@host# set ospf traffic-engineering
```

```
user@host# set ospf area 0.0.0.0 interface all
user@host# set ospf area 0.0.0.0 interface fxp0.0 disable
user@host# exit
```

5. Configure VPLS on PE1.

```
[edit]
user@host# edit routing-instances vpls
[edit routing-instances vpls]
user@host# set instance-type vpls
user@host# set interface ge-0/2/2.0
user@host# set route-distinguisher 65000:1
user@host# set vrf-target target:65000:1
user@host# set protocols vpls site-range 3
user@host# set protocols vpls no-tunnel-services
user@host# set protocols vpls site asia site-identifier 1
user@host# set protocols vpls site asia interface ge-0/2/2.0
user@host# set protocols vpls vpls-id 100
user@host# exit
```

Step-by-Step Procedure

To configure PE2:

1. Configure PE2's interfaces.

```
[edit]
user@host# edit interfaces
[edit interfaces]
user@host# set ge-0/0/7 gigether-options 802.3ad ae0
user@host# set ge-0/1/7 gigether-options 802.3ad ae0
user@host# set ge-0/2/3 gigether-options 802.3ad ae0
user@host# set ge-0/2/7 gigether-options 802.3ad ae0
user@host# set ge-2/0/0 gigether-options 802.3ad ae1
user@host# set ge-2/0/1 gigether-options 802.3ad ae1
user@host# set ge-2/0/2 gigether-options 802.3ad ae1
user@host# set ge-2/0/4 encapsulation ethernet-vpls
user@host# set ge-2/0/4 unit 0 family vpls
user@host# set ge-2/0/7 gigether-options 802.3ad ae0
user@host# set ge-2/0/9 unit 0 family inet address 1.1.1.1/30
user@host# set ge-2/0/9 unit 0 family mpls
user@host# set ge-2/1/7 gigether-options 802.3ad ae0
user@host# set ge-2/2/7 gigether-options 802.3ad ae0
user@host# set ge-2/3/7 gigether-options 802.3ad ae0
user@host# set ae0 unit 0 family inet address 11.11.11.2/30
user@host# set ae0 unit 0 family iso
user@host# set ae0 unit 0 family mpls
user@host# set ae1 unit 0 family inet address 10.1.1.1/30
user@host# set ae1 unit 0 family mpls
user@host# exit
```

2. Configure PE1's routing protocols and MPLS.

```
[edit]
user@host# edit protocols
[edit protocols]
user@host# set rsvp interface all
user@host# set rsvp interface fxp0.0 disable
user@host# set mpls label-switched-path PE2-to-PE3 from 10.255.19.77
```

```
user@host# set mpls label-switched-path PE2-to-PE3 to 10.255.19.79
user@host# set mpls label-switched-path PE2-to-PE1 to 10.255.71.214
user@host# set mpls interface all
user@host# set mpls interface fxp0.0 disable
user@host# set bgp group int type internal
user@host# set bgp group int local-address 10.255.19.77
user@host# set bgp group int family inet any
user@host# set bgp group int family l2vpn signaling
user@host# set bgp group int neighbor 10.255.71.214
user@host# set bgp group int neighbor 10.255.19.79
user@host# set ospf traffic-engineering
user@host# set ospf area 0.0.0.0 interface lo0.0
user@host# set ospf area 0.0.0.0 interface ge-2/0/0.0
user@host# set ospf area 0.0.0.0 interface ge-2/0/1.0
user@host# set ospf area 0.0.0.0 interface ge-2/0/2.0
user@host# set ospf area 0.0.0.0 interface ae0.0
user@host# set ospf area 0.0.0.0 interface ae1.0
user@host# set ospf area 0.0.0.0 interface all
user@host# set ospf area 0.0.0.0 interface fxp0.0 disable
user@host# set ldp interface all
user@host# set ldp interface fxp0.0 disable
user@host# exit
```

3. Configure VPLS on PE1.

```
[edit]
user@host# edit routing-instances vpls
[edit routing-instances vpls]
user@host# set instance-type vpls
user@host# set interface ge-2/0/4.0
user@host# set route-distinguisher 65000:1
user@host# set vrf-target target:65000:1
user@host# set protocols vpls site-range 3
user@host# set protocols vpls no-tunnel-services
user@host# set protocols vpls site 2 site-identifier 2
user@host# set protocols vpls site 2 interface ge-2/0/4.0
user@host# set protocols vpls vpls-id 100
user@host# exit
```

Results From configuration mode, confirm your configuration by entering the **show forwarding-options**, **show interfaces**, **show policy-options**, **show protocols**, and **show routing-instances** commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

PE1:

```
user@host# show forwarding-options
hash-key {
   family multiservice {
      source-mac;
      destination-mac;
      payload {
         ip {
            layer-3;
         }
```



```
    }
  }
}
user@host# show interfaces
ge-0/0/6 {
  gigether-options {
    802.3ad ae0;
  }
}
ge-0/1/6 {
  gigether-options {
    802.3ad ae0;
  }
}
ge-0/2/2 {
  encapsulation ethernet-vpls;
  unit 0 {
    family vpls;
  }
}
ge-0/2/3 {
  gigether-options {
    802.3ad ae0;
  }
}
ge-0/2/6 {
  gigether-options {
    802.3ad ae0;
  }
}
ge-0/3/0 {
  gigether-options {
    802.3ad ae0;
  }
}
ge-0/3/1 {
  gigether-options {
    802.3ad ae0;
  }
}
ge-0/3/6 {
  gigether-options {
    802.3ad ae0;
  }
}
ge-1/0/6 {
  gigether-options {
    802.3ad ae0;
  }
}
ge-1/2/6 {
  unit 0 {
    family inet {
      address 13.1.1.2/30;
    }
```

```
    }
  }
  ae0 {
    unit 0 {
      family inet {
        address 11.11.11.1/30;
      }
      family iso;
      family mpls;
    }
  }
```

```
user@host# show policy-options
policy-statement exp-to-fwd {
  term a {
    from community grn-com;
    then {
      install-nexthop lsp PE1-to-PE2;
      accept;
    }
  }
}
community grn-com members target:65000:1;
```

```
user@host# show protocols
mpls {
  label-switched-path PE1-to-PE2 {
    to 10.255.19.77;
  }
  label-switched-path PE1-to-PE3 {
    to 10.255.19.79;
  }
}
bgp {
  group int {
    type internal;
    local-address 10.255.71.214;
    family inet {
      any;
    }
    family l2vpn {
      signaling;
    }
    neighbor 10.255.19.77;
    neighbor 10.255.19.79;
  }
}
ospf {
  traffic-engineering;
  area 0.0.0.0 {
    interface all;
    interface fxp0.0 {
      disable;
    }
  }
}
```

```
user@host# show routing-instances
vpls {
    instance-type vpls;
    interface ge-0/2/2.0;
    route-distinguisher 65000:1;
    vrf-target target:65000:1;
    protocols {
        vpls {
            site-range 3;
            no-tunnel-services;
            site asia {
                site-identifier 1;
                interface ge-0/2/2.0;
            }
            vpls-id 100;
        }
    }
}
```

PE2:

```
user@host# show interfaces
ge-0/0/7 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-0/1/7 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-0/2/3 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-0/2/7 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-2/0/0 {
    gigether-options {
        802.3ad ae1;
    }
}
ge-2/0/1 {
    gigether-options {
        802.3ad ae1;
    }
}
ge-2/0/2 {
    gigether-options {
        802.3ad ae1;
    }
```

```
}
ge-2/0/4 {
   encapsulation ethernet-vpls;
   unit 0 {
      family vpls;
   }
}
ge-2/0/7 {
   gigether-options {
      802.3ad ae0;
   }
}
ge-2/0/9 {
   unit 0 {
      family inet {
         address 1.1.1.1/30;
      }
      family mpls;
   }
}
ge-2/1/7 {
   gigether-options {
      802.3ad ae0;
   }
}
ge-2/2/7 {
   gigether-options {
      802.3ad ae0;
   }
}
ge-2/3/7 {
   gigether-options {
      802.3ad ae0;
   }
}
ae0 {
   unit 0 {
      family inet {
         address 11.11.11.2/30;
      }
      family iso;
      family mpls;
   }
}
ae1 {
   unit 0 {
      family inet {
         address 10.1.1.1/30;
      }
      family mpls;
   }
}
```

user@host# **show protocols**

```
rsvp {
   interface all;
```

```
    interface fxp0.0 {
      disable;
    }
  }
  mpls {
    label-switched-path PE2-to-PE3 {
      from 10.255.19.77;
      to 10.255.19.79;
    }
    label-switched-path PE2-to-PE1 {
      to 10.255.71.214;
    }
    interface all;
    interface fxp0.0 {
      disable;
    }
  }
  bgp {
    group int {
      type internal;
      local-address 10.255.19.77;
      family inet {
        any;
      }
      family l2vpn {
        signaling;
      }
      neighbor 10.255.71.214;
      neighbor 10.255.19.79;
    }
  }
  ospf {
    traffic-engineering;
    area 0.0.0.0 {
      interface lo0.0;
      interface ge-2/0/0.0;
      interface ge-2/0/1.0;
      interface ge-2/0/2.0;
      interface ae0.0;
      interface ae1.0;
      interface all;
      interface fxp0.0 {
        disable;
      }
    }
  }
  ldp {
    interface all;
    interface fxp0.0 {
      disable;
    }
  }
```

```
user@host# show routing-instances
vpls {
    instance-type vpls;
```

```
    interface ge-2/0/4.0; ## 'ge-2/0/4.0' is not defined
    route-distinguisher 65000:1;
    vrf-target target:65000:1;
    protocols {
      vpls {
        site-range 3;
        no-tunnel-services;
        site 2 {
          site-identifier 2;
          interface ge-2/0/4.0;
        }
        vpls-id 100;
      }
    }
  }
```

Verification

You can monitor the operation of the routing instance by running the **show interfaces ae1.0 extensive** and **monitor interface traffic** commands.

For troubleshooting, you can configure tracing operations for all of the protocols.

Related Documentation

- Configuring Point-to-Multipoint LSPs for an MBGP MVPN
- Understanding Wildcards to Configure Selective Point-to-Multipoint LSPs for an MBGP MVPN
- show interfaces (Aggregated Ethernet)
- *Junos OS Multicast over Layer 3 VPNs Feature Guide*

CHAPTER 11

Traffic Sampling Configuration

Traffic Sampling Configuration

To configure traffic sampling, include the **sampling** statement at the **[edit forwarding-options]** hierarchy level:

```
[edit forwarding-options]
sampling {
    disable;
    family (inet | inet6 | mpls) {
        disable;
        output {
            aggregate-export-interval seconds;
            extension-service service-name;
            file {
                disable;
                filename filename;
                files number;
                size bytes;
                (stamp | no-stamp);
                (world-readable | no-world-readable);
            }
            flow-active-timeout seconds;
            flow-inactive-timeout seconds;
```

```
        flow-server hostname {
          aggregation {
            autonomous-system;
            destination-prefix;
            protocol-port;
            source-destination-prefix {
              caida-compliant;
            }
            source-prefix;
          }
          autonomous-system-type (origin | peer);
          (local-dump | no-local-dump);
          port port-number;
          source-address address;
          version format;
          version9 {
            template template-name;
          }
        }
        interface interface-name {
          engine-id number;
          engine-type number;
          source-address address;
        }
      }
    }
    input {
      max-packets-per-second number;
      maximum-packet-length bytes;
      rate number;
      run-length number;
    }
    traceoptions {
      file filename {
        files number;
        size bytes;
        (world-readable | no-world-readable);
      }
    }
  }
```

Minimum Traffic Sampling Configuration

To configure traffic sampling, you must perform at least the following tasks:

1. Create a firewall filter to apply to the logical interfaces being sampled by including the **filter** statement at the **[edit firewall family *family-name*]** hierarchy level. In the filter **then** statement, you must specify the action modifier **sample** and the action **accept**.

   ```
   [edit firewall family family-name]
   filter filter-name {
     term term-name {
       then {
         sample;
         accept;
   ```

```
    }
  }
}
```

2. Apply the filter to the interfaces on which you want to sample traffic:

```
[edit interfaces]
interface-name {
  unit logical-unit-number {
    family family-name {
      filter {
        input filter-name;
      }
      address address {
        destination destination-address;
      }
    }
  }
}
```

3. Enable sampling and specify a nonzero sampling rate:

```
[edit forwarding-options]
sampling {
  input {
    rate number;
  }
}
```

Configuring Traffic Sampling

On routing platforms containing a Monitoring Services PIC or an Adaptive Services PIC, you can configure traffic sampling for traffic passing through the routing platform.

To configure traffic sampling on a logical interface, include the **input** statement at the **[edit forwarding-options sampling]** hierarchy level:

```
[edit forwarding-options sampling]
input {
  max-packets-per-second number;
  maximum-packet-length bytes;
  rate number;
  run-length number;
}
```

In Junos OS Release 8.3 and later, you can also configure traffic sampling of MPLS traffic.

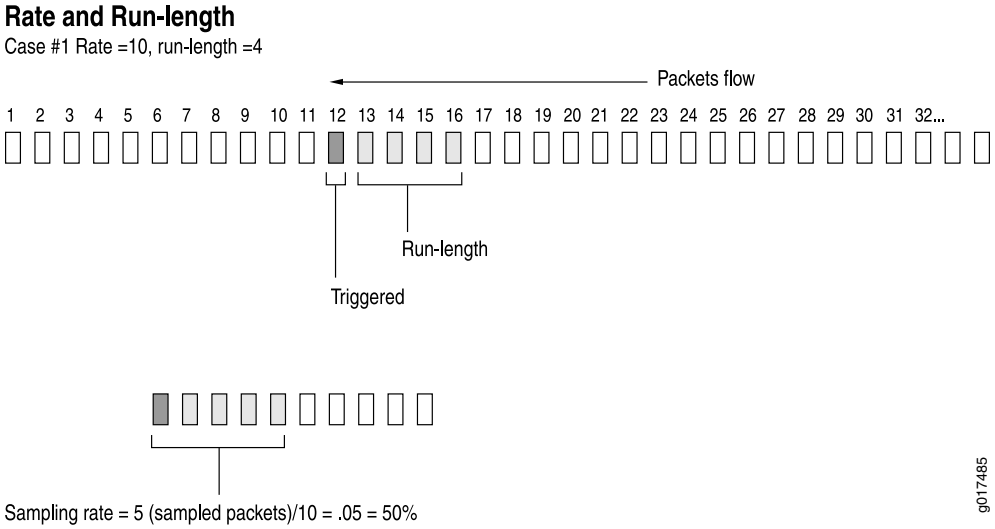
Specify the threshold traffic value by using the **max-packets-per-second** statement. The value is the maximum number of packets to be sampled, beyond which the sampling mechanism begins dropping packets. The range is 0 through 65,535. A value of 0 instructs the Packet Forwarding Engine not to sample any packets. The default value is 1000.

NOTE: This statement is not valid for port mirroring.

Specify the maximum length of the sampled packet by using the **maximum-packet-length** ***bytes*** statement. For ***bytes***, specify a value from 0 through 9192.

Specify the sampling rate by setting the values for **rate** and **run-length** (see Figure 15 on page 186).

Figure 15: Configure Sampling Rate



The forwarding plane provides support for random sampling that can be configured through the **rate** or **run-length** statement. The **rate** statement sets the ratio of the number of packets to be sampled on an average. For example, if you configure a rate of 10, on average every tenth packet (1 packet out of 10) is sampled.

The **run-length** statement specifies the number of matching packets to sample following the initial one-packet trigger event. Configuring a run length greater than 0 allows you to sample packets following those already being sampled.

NOTE: **The run-length and maximum-packet-length configuration statements are not supported on MX80 routers.**

You can also send the sampled packets to a specified host using the cflowd version 5 and 8 formats or the version 9 format as defined in RFC 3954. For more information, see "Configuring Flow Aggregation (cflowd)" on page 189 and "Configuring Active Flow Monitoring Using Version 9" on page 192.

The Junos OS does not sample packets originating from the router. If you configure a sampling filter and apply it to the output side of an interface, then only the transit packets going through that interface are sampled. Packets that are sent from the Routing Engine to the Packet Forwarding Engine are not sampled.

When you apply a firewall filter to a loopback interface, the filter might block responses from the Monitoring Services PIC. To allow responses from the Monitoring Services PIC

to pass through for sampling purposes, configure a term in the firewall filter to include the Monitoring Services PIC's IP address.

Related Documentation

- Guidelines for Configuring Standard Firewall Filters
- Guidelines for Applying Standard Firewall Filters

Disabling Traffic Sampling

To explicitly disable traffic sampling on the router, include the **disable** statement at the **[edit forwarding-options sampling]** hierarchy level:

```
[edit forwarding-options sampling]
disable;
```

Configuring the Output File for Traffic Sampling

You configure traffic sampling results to a file in the **/var/tmp** directory. To collect the sampled packets in a file, include the **file** statement at the **[edit forwarding-options sampling output]** hierarchy level:

```
[edit forwarding-options sampling family family-name output]
file <disable> filename filename <files number> <size bytes> <stamp | no-stamp >
   <world-readable | no-world-readable>;
```

To configure the period of time before an active flow is exported, include the **flow-active-timeout** statement at the **[edit forwarding-options sampling output family (inet | inet6 | mpls)]** hierarchy level:

```
[edit forwarding-options sampling family (inet | inet6 | mpls) output]
flow-active-timeout seconds;
```

To configure the period of time before a flow is considered inactive, include the **flow-inactive-timeout** statement at the **[edit forwarding-options sampling output]** hierarchy level:

```
[edit forwarding-options sampling family (inet | inet6 | mpls) output]
flow-inactive-timeout seconds;
```

To configure the interface that sends out monitored information, include the **interface** statement at the **[edit forwarding-options sampling output]** hierarchy level:

```
[edit forwarding-options sampling family (inet | inet6 | mpls) output]
interface interface-name {
   engine-id number;
   engine-type number;
   source-address address;
}
```

NOTE: This feature is not supported with the version 9 template format. You must send traffic flows collected using version 9 to a server. For more information see "Configuring Active Flow Monitoring Using Version 9" on page 192.

Traffic Sampling Output Format

Traffic sampling output is saved to an ASCII text file. The following is an example of the traffic sampling output that is saved to a file in the **/var/tmp** directory. Each line in the output file contains information for one sampled packet. You can optionally display a timestamp for each line.

The column headers are repeated after each group of 1000 packets.

```
# Apr  7 15:48:50
Time                   Dest           Src Dest Src Proto TOS Pkt Intf  IP   TCP

                       addr          addr port port          len num frag flags
Apr 7 15:48:54 192.168.9.194 192.168.9.195   0    0   1   0x0  84  8   0x0   0x0
Apr 7 15:48:55 192.168.9.194 192.168.9.195   0    0   1   0x0  84  8   0x0   0x0
Apr 7 15:48:56 192.168.9.194 192.168.9.195   0    0   1   0x0  84  8   0x0   0x0
Apr 7 15:48:57 192.168.9.194 192.168.9.195   0    0   1   0x0  84  8   0x0   0x0
Apr 7 15:48:58 192.168.9.194 192.168.9.195   0    0   1   0x0  84  8   0x0   0x0
```

The output contains the following fields:

- **Time**—Time at which the packet was received (displayed only if you include the **stamp** statement in the configuration)
- **Dest addr**—Destination IP address in the packet
- **Src addr**—Source IP address in the packet
- **Dest port**—Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) port for the destination address
- **Src port**—TCP or UDP port for the source address
- **Proto**—Packet's protocol type
- **TOS**—Contents of the type-of-service (ToS) field in the IP header
- **Pkt len**—Length of the sampled packet, in bytes
- **Intf num**—Unique number that identifies the sampled logical interface
- **IP frag**—IP fragment number, if applicable
- **TCP flags**—Any TCP flags found in the IP header

To set the timestamp option for the file **my-sample**, enter the following:

```
[edit forwarding-options sampling family (inet | inet6 | mpls) output file]
user@host# set filename my-sample files 5 size 2m world-readable stamp;
```

Whenever you toggle the timestamp option, a new header is included in the file. If you set the **stamp** option, the **Time** field is displayed.

```
# Apr  7 15:48:50
# Time            Dest        Src  Dest   Src Proto  TOS   Pkt  Intf    IP   TCP
#                 addr       addr  port  port             len   num  frag flags
# Feb  1 20:31:21
#                 Dest        Src  Dest   Src Proto  TOS   Pkt  Intf    IP   TCP
#                 addr       addr  port  port             len   num  frag flags
```

Tracing Traffic-Sampling Operations

Tracing operations track all traffic-sampling operations and record them in a log file in the **/var/log** directory. By default, this file is named **/var/log/sampled**. The default file size is 128 KB, and 10 files are created before the first one gets overwritten.

To trace traffic-sampling operations, include the **traceoptions** statement at the **[edit forwarding-options sampling]** hierarchy level:

```
[edit forwarding-options sampling]
traceoptions {
  file <filename> <files number> <size bytes> <world-readable | no-world-readable>;
  no-remote-trace;
}
```

Configuring Flow Aggregation (cflowd)

You can collect an aggregate of sampled flows and send the aggregate to a specified host that runs the cflowd application available from the Cooperative Association for Internet Data Analysis (CAIDA) (**http://www.caida.org**). By using cflowd, you can obtain various types of byte and packet counts of flows through a router.

The cflowd application collects the sampled flows over a period of 1 minute. At the end of the minute, the number of samples to be exported are divided over the period of another minute and are exported over the course of the same minute.

Before you can perform flow aggregation, the routing protocol process must export the autonomous system (AS) path and routing information to the sampling process. To do this, include the **route-record** statement:

```
route-record;
```

You can include this statement at the following hierarchy levels:

- **[edit routing-options]**
- **[edit routing-instances** *routing-instance-name* **routing-options]**

By default, flow aggregation is disabled. To enable the collection of flow aggregates, include the **flow-server** statement at the **[edit forwarding-options sampling output]** hierarchy level:

```
[edit forwarding-options sampling family (inet | inet6 | mpls) output ]
flow-server hostname {
  aggregation {
    autonomous-system;
    destination-prefix;
    protocol-port;
    source-destination-prefix {
      caida-compliant;
    }
    source-prefix;
  }
```

```
    autonomous-system-type (origin | peer);
    (local-dump | no-local-dump);
    port port-number;
    source-address address;
    version format;
}
```

In the cflowd statement, specify the name, identifier, and source-address of the host that collects the flow aggregates. You must also include the UDP port number on the host and the **version**, which gives the format of the exported cflowd aggregates. To specify an IPv4 source address, include the **source-address** statement. To collect cflowd records in a log file before exporting, include the **local-dump** statement. To specify the cflowd version number, include the **version** statement. The cflowd version is either 5 or 8.

You can specify both host (cflowd) sampling and port mirroring in the same configuration. You can perform RE-sampling and port mirroring actions simultaneously. However, you cannot perform PIC-sampling and port mirroring actions simultaneously.

To specify aggregation of specific types of traffic, include the **aggregation** statement. This conserves memory and bandwidth enabling cflowd to export targeted flows rather than all the aggregated

NOTE: **Aggregation is valid only if cflowd version 8 is specified.**

To specify a flow type, include the **aggregation** statement at the **[edit forwarding-options sampling output cflowd *hostname*]** hierarchy level:

```
[edit forwarding-options sampling family (inet | inet6 | mpls) output hostname]
aggregation {
    source-destination-prefix;
}
```

You specify the aggregation type using one of the following options:

- **autonomous-system**—Aggregate by AS number; may require setting the separate cflowd **autonomous-system-type** statement to include either **origin** or **peer** AS numbers. The **origin** option specifies to use the origin AS of the packet source address in the Source Autonomous System cflowd field. The **peer** option specifies to use the peer AS through which the packet passed in the Source Autonomous System cflowd field. By default, **cflowd** exports the origin AS number.
- **destination-prefix**—Aggregate by destination prefix (only).
- **protocol-port**—Aggregate by protocol and port number; requires setting the separate **cflowd port** statement.
- **source-destination-prefix**—Aggregate by source and destination prefix. Version 2.1b1 of CAIDA's cflowd application does not record source and destination mask length values in compliance with CAIDA's *cflowd Configuration Guide*, dated August 30, 1999. If you configure the **caida-compliant** statement, the Junos OS complies with Version 2.1b1 of cflowd. If you do not include the **caida-compliant** statement in the configuration,

the Junos OS records source and destination mask length values in compliance with the *cflowd Configuration Guide*.

- **source-prefix**—Aggregate by source prefix (only).

Collection of sampled packets in a local ASCII file is not affected by the **cflowd** statement.

Debugging cflowd Flow Aggregation

To collect the cflowd flows in a log file before they are exported, include the **local-dump** option at the **[edit forwarding-options sampling output cflowd *hostname*]** hierarchy level:

```
[edit forwarding-options sampling family (inet | inet6 | mpls) output flow-server hostname]
local-dump;
```

By default, the flows are collected in **/var/log/sampled**; to change the filename, include the **filename** statement at the **[edit forwarding-options sampling traceoptions]** hierarchy level. For more information about changing the filename, see "Configuring the Output File for Traffic Sampling" on page 187.

NOTE: Because the **local-dump** option adds extra overhead, you should use it only while debugging cflowd problems, not during normal operation.

The following is an example of the flow information. The AS number exported is the origin AS number. All flows that belong under a cflowd header are dumped, followed by the header itself:

```
Jun 27 18:35:43 v5 flow entry
Jun 27 18:35:43    Src addr: 10.53.127.1
Jun 27 18:35:43    Dst addr: 10.6.255.15
Jun 27 18:35:43    Nhop addr: 192.168.255.240
Jun 27 18:35:43    Input interface: 5
Jun 27 18:35:43    Output interface: 3
Jun 27 18:35:43    Pkts in flow: 15
Jun 27 18:35:43    Bytes in flow: 600
Jun 27 18:35:43    Start time of flow: 7230
Jun 27 18:35:43    End time of flow: 7271
Jun 27 18:35:43    Src port: 26629
Jun 27 18:35:43    Dst port: 179
Jun 27 18:35:43    TCP flags: 0x10
Jun 27 18:35:43    IP proto num: 6
Jun 27 18:35:43    TOS: 0xc0
Jun 27 18:35:43    Src AS: 7018
Jun 27 18:35:43    Dst AS: 11111
Jun 27 18:35:43    Src netmask len: 16
Jun 27 18:35:43    Dst netmask len: 0
```

[... 41 more **v5 flow** entries; then the following header:]

```
Jun 27 18:35:43 cflowd header:
Jun 27 18:35:43   Num-records: 42
Jun 27 18:35:43   Version: 5
Jun 27 18:35:43   Flow seq num: 118
Jun 27 18:35:43   Engine id: 0
Jun 27 18:35:43   Engine type: 3
```

Configuring Active Flow Monitoring Using Version 9

In Junos OS Release 8.3 and later, you can collect a record of sampled flows using the version 9 format as defined in RFC 3954, *Cisco Systems NetFlow Services Export Version 9*. Version 9 uses templates to collect an set of sampled flows and send the record to a specified host.

You configure the version 9 template used to collect a record of sampled flows at the **[edit services monitoring]** hierarchy level. For more information, see the *Junos OS Services Interfaces Configuration Guide* and the *Junos OS Flow Monitoring Feature Guide*.

To enable the collection of traffic flows using the version 9 format, include the **version9** statement at the **[edit forwarding-options sampling family *family-name* output flow-server *hostname*]** hierarchy level:

```
[edit forwarding-options sampling family family-name output flow-server hostname]
version9 {
  template template-name;
}
```

template-name is the name of the version 9 template configured at the **[edit services monitoring]** hierarchy level.

You configure traffic sampling at the **[edit forwarding-options sampling input]** hierarchy level. In Junos OS Release 8.3 and later, you can configure sampling for MPLS traffic as well as IPv4 traffic. You can define a version 9 flow record template suitable for IPv4 traffic, MPLS traffic, or a combination of the two. In Junos OS Release 9.5 and later, you can sample packets from both the **inet** and **mpls** protocol families at the same time. In Junos OS Release 10.4 and later, you can configure sampling for peer AS billing traffic for the **inet** and **ipv6** protocols only. For more information about how to configure traffic sampling, see "Configuring Traffic Sampling" on page 185.

The following restrictions apply to configuration of the version 9 format:

- You can configure only one host to collect traffic flows using the version 9 format. Configure the host at the **[edit forwarding-options sampling family *family-name* output flow-server *hostname*]** hierarchy level.
- You cannot specify both the version 9 format and cflowd versions 5 and 8 formats in the same configuration. For more information about how to configure flow monitoring using cflowd version 8, see "Configuring Flow Aggregation (cflowd)" on page 189.
- Any values for **flow-active-timeout** and **flow-inactive-timeout** that you configure at the **[edit forwarding-options sampling output]** hierarchy level are overridden by the values configured in the version 9 template.
- Version 9 does not support Routing Engine-based sampling. You cannot configure version 9 to send traffic sampling result to a file in the **/var/tmp** directory.

Example: Configuring Active Flow Monitoring Using Version 9

In this example, you enable active flow monitoring using version 9. You specify a template **mpls** that you configure at the **[edit services monitoring]** hierarchy level. You also configure the traffic family **mpls** to sample MPLS packets.

```
[edit forwarding-options]
sampling {
  input {
    rate 1;
    run-length;
  }
  family inet {
    output {
      flow-server 10.60.2.1 { # The IP address and port of the host
        port 2055; # that collects the sampled traffic flows.
        source-address 3.3.3.1;
        version9 {
          template mpls; # Version 9 records are sent
        } # using the template named mpls
      }
    }
  }
}
```

Traffic Sampling Examples

The following sections provide examples of configuring traffic sampling:

- Example: Sampling a Single SONET/SDH Interface on page 193
- Example: Sampling All Traffic from a Single IP Address on page 194
- Example: Sampling All FTP Traffic on page 195

Example: Sampling a Single SONET/SDH Interface

The following configuration gathers statistical sampling information from a small percentage of all traffic on a single SONET/SDH interface and collects it in a file named **sonet-samples.txt**.

Create the filter:

```
[edit firewall family inet]
filter {
  sample-sonet {
    then {
      sample;
      accept;
    }
  }
}
```

Apply the filter to the SONET/SDH interface:

```
[edit interfaces]
so-0/0/1 {
    unit 0 {
        family inet {
            filter {
                input sample-sonet;
            }
            address 10.127.68.254/32 {
                destination 10.127.74.7;
            }
        }
    }
}
```

Finally, configure traffic sampling:

```
[edit forwarding-options]
sampling {
    input {
        rate 100;
        run-length 2;
    }
    family inet {
        output {
            file {
                filename sonet-samples.txt;
                files 40;
                size 5m;
            }
        }
}
```

Example: Sampling All Traffic from a Single IP Address

The following configuration gathers statistical information about every packet entering the router on a specific Gigabit Ethernet port originating from a single source IP address of **10.45.92.31**, and collects it in a file named **samples-10-45-92-31.txt**.

Create the filter:

```
[edit firewall family inet]
filter one-ip {
    term get-ip {
        from {
            source-address 10.45.92.31;
        }
        then {
            sample;
            accept;
        }
    }
}
```

Apply the filter to the Gigabit Ethernet interface:

```
[edit interfaces]
```

```
ge-4/1/1 {
  unit 0 {
    family inet {
      filter {
        input one-ip;
      }
      address 10.45.92.254;
    }
  }
}
```

Finally, gather statistics on all the candidate samples; in this case, gather all statistics:

```
[edit forwarding-options]
sampling {
  input {
    rate 1;
  }
  family inet {
    output {
      file {
        filename samples-215-45-92-31.txt;
        files 100;
        size 100k;
      }
    }
  }
}
```

Example: Sampling All FTP Traffic

The following configuration gathers statistical information about a moderate percentage of packets using FTP in the output path of a specific T3 interface, and collects the information in a file named **t3-ftp-traffic.txt**.

Create a filter:

```
[edit firewall family inet]
filter ftp-stats {
  term ftp-usage {
    from {
      destination-port [ftp ftp-data];
    }
    then {
      sample;
      accept;
    }
  }
}
```

Apply the filter to the T3 interface:

```
[edit interfaces]
t3-7/0/2 {
  unit 0 {
    family inet {
```

```
        filter {
            input ftp-stats;
        }
        address 10.35.78.254/32 {
            destination 10.35.78.4;
        }
    }
  }
}
```

Finally, gather statistics on 10 percent of the candidate samples:

```
[edit forwarding-options]
sampling {
  input {
    rate 10;
  }
  family inet {
    output {
      file {
        filename t3-ftp-traffic.txt;
        files 50;
        size 1m;
      }
    }
  }
}
```

CHAPTER 12

Traffic Forwarding and Monitoring Configuration

Configuring Traffic Forwarding and Monitoring

To configure forwarding options and traffic monitoring, include statements at the **[edit forwarding-options]** hierarchy level:

```
[edit forwarding-options]
accounting group-name {
  output {
    cflowd [ hostnames ] {
      aggregation {
        autonomous-system;
        destination-prefix;
        protocol-port;
        source-destination-prefix {
          caida-compliant;
```

```
            }
            source-prefix;
          }
          autonomous-system-type (origin | peer);
          port port-number;
          version format;
        }
        flow-active-timeout seconds;
        flow-inactive-timeout seconds;
        interface interface-name {
          engine-id number;
          engine-type number;
          source-address address;
        }
      }
    }
    enhanced-hash-key {
      family [
        inet
          incoming-interface-index;
            no-destination-port;
            no-source-port;
            type-of-service;
          }
        inet6 {
          incoming-interface-index;
          no-destination-port;
          no-source-port;
          traffic-class;
        }
        mpls {
          incoming-interface-index;
          label-1-exp;
          no-payload;
        }
        multiservice {
          incoming-interface-index;
          no-payload;
          outer-priority;
        }
      }
      services-loadbalancing {
        family inet layer-3-services {
          incoming-interface-index;
          source-address;
        }
      }
    }
    family family-name {
      filter {
        input filter-name;
        output filter-name;
      }
      route-accounting;
    }
    flood {
```

```
    input filter-name;
}
hash-key {
    family inet {
        layer-3;
        layer-4;
    }
    family mpls {
        no-interface-index;
        label-1;
        label-2;
        label-3;
        no-labels;
        no-label-1-exp;
        payload {
            ether-pseudowire;
            ip {
                layer-3-only;
                port-data {
                    source-msb;
                    source-lsb;
                    destination-msb;
                    destination-lsb;
                }
            }
        }
    }
    family multiservice }
        destination-mac;
        label-1;
        label-2;
        payload {
            ip {
                layer-3-only;
            }
        }
        source-mac;
    }
}
helpers {
    bootp {
        client-response-ttl;
        description text-description;
        interface interface-group {
            client-response-ttl number;
            description text-description;
            maximum-hop-count number;
            minimum-wait-time seconds;
            no-listen;
            server address {
                logical-system logical-system-name <routing-instance [ <default>
                    routing-instance-names ]>;
                routing-instance [ <default> routing-instance-names ];
            }
        }
        maximum-hop-count number;
```

```
        minimum-wait-time seconds;
        relay-agent-option;
        server [ addresses ];
    }
    domain {
        description text-description;
        server < [ routing-instance routing-instance-names ] >;
        interface interface-name {
            description text-description;
            no-listen;
            server < [ routing-instance routing-instance-names ] >;
        }
    }
    tftp {
        description text-description;
        server < [ routing-instance routing-instance-names ] >;
        interface interface-name {
            description text-description;
            no-listen;
            server < [ routing-instance routing-instance-names ] >;
        }
    }
    traceoptions {
        file <filename> <files number> <match regular-expression> <size size>
            <world-readable | no-world readable>;
        flag flag;
        level severity-level;
        no-remote-trace;
    }
}
load-balance {
    indexed-load-balance;
    per-flow {
        hash-seed number;
    }
    per-prefix {
        hash-seed number;
    }
}
monitoring group-name {
    family inet {
        output {
            cflowd hostname {
                port port-number;
            }
            export-format cflowd-version-5;
            flow-active-timeout seconds;
            flow-export-destination {
                cflowd-collector;
            }
            flow-inactive-timeout seconds;
            interfaceinterface-name {
                engine-id number;
                engine-type number;
                input-interface-index number;
                output-interface-index number;
```

```
            source-address address;
          }
        }
      }
    }
    next-hop-group [ group-names ] {
      interface interface-name {
        next-hop [ addresses ];
      }
    }
    packet-capture {
      disable;
      file filename file-name <files number> <size number> <world-readable |
        no-world-readable>;
      maximum-capture-size bytes;
    }
    port-mirroring {
      family (ccc | inet | inet6 | vpls) {
        output {
          interface interface-name {
            next-hop address;
          }
          no-filter-check;
        }
        input {
          maximum-packet-length bytes;
          rate number;
          run-length number;
        }
      }
      traceoptions {
        file <filename> <files number> <match regular-expression> <size bytes>
          <world-readable | no-world-readable>;
        no-remote-trace;
      }
    }
```

Configuring Forwarding Table Filters

The following sections describe the following topics:

- Overview of Forwarding Table Filters on page 201
- Configuring a Forwarding Table Filter on page 202

Overview of Forwarding Table Filters

Forwarding table filters are defined the same as other firewall filters, but you apply them differently:

- Instead of applying forwarding table filters to interfaces, you apply them to forwarding tables, each of which is associated with a routing instance and a virtual private network (VPN).
- Instead of applying input and output filters by default, you can apply an input forwarding table filter only.

All packets are subjected to the input forwarding table filter that applies to the forwarding table. A forwarding table filter controls which packets the router accepts and then performs a lookup for the forwarding table, thereby controlling which packets the router forwards on the interfaces.

When the router receives a packet, it determines the best route to the ultimate destination by looking in a forwarding table, which is associated with the VPN on which the packet is to be sent. The router then forwards the packet toward its destination through the appropriate interface.

NOTE: For transit packets exiting the router through the tunnel, forwarding table filtering is not supported on the interfaces you configure as the output interface for tunnel traffic.

Configuring a Forwarding Table Filter

A forwarding table filter allows you to filter data packets based on their components and to perform an action on packets that match the filter; it essentially controls which bearer packets the router accepts and forwards. To configure a forwarding table filter, include the **firewall** statement at the **[edit]** hierarchy level:

```
[edit]
firewall {
   family family-name {
      filter filter-name {
         term term-name {
            from {
               match-conditions;
            }
            then {
               action;
               action-modifiers;
            }
         }
      }
   }
}
```

family-name is the family address type: IPv4 (**inet**), IPv6 (**inet6**), Layer 2 traffic (**bridge**), or MPLS (**mpls**).

term-name is a named structure in which match conditions and actions are defined.

match-conditions are the criteria against which a bearer packet is compared; for example, the IP address of a source device or a destination device. You can specify multiple criteria in a match condition.

action specifies what happens if a packet matches all criteria; for example, the gateway GPRS support node (GGSN) accepting the bearer packet, performing a lookup in the forwarding table, and forwarding the packet to its destination; discarding the packet; and discarding the packet and returning a rejection message.

action-modifiers are actions that are taken in addition to the GGSN accepting or discarding a packet when all criteria match; for example, counting the packets and logging a packet.

To create a forwarding table, include the **instance-type** statement with the **forwarding** option at the **[edit routing-instances *instance-name*]** hierarchy level:

```
[edit]
routing-instances instance-name {
   instance-type forwarding;
}
```

To apply a forwarding table filter to a VPN routing and forwarding (VRF) table, include the **filter** and **input** statements at the **[edit routing-instance *instance-name* forwarding-options family *family-name*]** hierarchy level:

```
[edit routing-instances instance-name]
instance-type forwarding;
forwarding-options {
   family family-name {
      filter {
         input filter-name;
      }
   }
}
```

To apply a forwarding table filter to a forwarding table, include the **filter** and **input** statements at the **[edit forwarding-options family *family-name*]** hierarchy level:

```
[edit forwarding-options family family-name]
filter {
   input filter-name;
}
```

To apply a forwarding table filter to the default forwarding table **inet.0**, which is not associated with a specific routing instance, include the **filter** and **input** statements at the **[edit forwarding-options family inet]** hierarchy level:

```
[edit forwarding-options family inet]
filter {
   input filter-name;
}
```

Related Documentation

- Guidelines for Configuring Standard Firewall Filters
- Guidelines for Applying Standard Firewall Filters
- Applying Filters to Forwarding Tables on page 203

Applying Filters to Forwarding Tables

A forwarding table filter allows you to filter data packets based on their components and perform an action on packets that match the filter. You can apply a filter on the ingress or egress packets of a forwarding table. You configure the filter at the **[edit firewall family *family-name*]** hierarchy level; for more information, see "Configuring Forwarding Table Filters" on page 201.

To apply a forwarding table filter on ingress packets of a forwarding table, include the **filter** and **input** statements at the **[edit forwarding-options family *family-name*]** hierarchy level:

```
[edit forwarding-options family family-name]
filter {
  input filter-name;
}
```

On the MX Series router only, to apply a forwarding table filter for a virtual switch, include the **filter** and **input** statements at the **[edit routing-instances *routing-instance-name* bridge-domains *bridge-domain-name* forwarding-options]** hierarchy level:

```
[edit routing-instances routing-instance-name bridge-domains bridge-domain-name
  forwarding-options]
  filter {
    input filter-name;
  }
```

For more information about how to configure a virtual switch, see the *Junos OS Layer 2 Configuration Guide*.

You can filter based upon destination-class information by applying a firewall filter on the egress packets of the forwarding table. By applying firewall filters to packets that have been forwarded by a routing table, you can match based on certain parameters that are decided by the route lookup. For example, routes can be classified into specific destination and source classes. Firewall filters used for policing and mirroring are able to match based upon these classes.

To apply a firewall filter on egress packets of a forwarding table, include the **filter** and **output** statements at the **[edit forwarding-options family *family-name*]** hierarchy level:

```
[edit forwarding-options family family-name]
filter {
  output filter-name;
}
```

NOTE: The egress forwarding table filter is applied on the ingress interface of the Flexible PIC Concentrator (FPC). If different packets to the same destination arrive on different FPCs, they might encounter different policers.

NOTE: You cannot simultaneously include the **interface-group** statement at the **[edit firewall family inet filter *filter-name* term *term-name* from]** hierarchy level and configure an egress forwarding table filter. The egress forwarding table filter is applied to transit packets only.

NOTE: The egress forwarding table filter is not supported for the J Series Services Routers.

NOTE: In Junos OS Release 8.4 and later, you can no longer configure this output statement for VPLS. You can continue to configure ingress forwarding table filters with the input statement at the **[edit forwarding-options family vpls filter]** hierarchy level.

To apply a forwarding table filter to a flood table, include the **flood** and **input** statements at the **[edit forwarding-options family *family-name*]** hierarchy level:

```
[edit forwarding-options family vpls]
flood {
  input filter-name;
}
```

NOTE: The **flood** statement is valid for the **vpls** protocol family only.

Configuring IPv6 Accounting

You can configure the routing platform to track IPv6-specific packets and bytes passing through the router.

To enable IPv6 accounting, include the **route-accounting** statement at the **[edit forwarding-options family inet6]** hierarchy level:

```
[edit forwarding-options family inet6]
route-accounting;
```

By default, IPv6 accounting is disabled. If IPv6 accounting is enabled, it remains enabled after a reboot of the router. To view IPv6 statistics, issue the **show interface statistics** operational mode command.

Related Documentation

- *Junos OS Interfaces Command Reference*

Configuring Discard Accounting

On routing platforms containing a Monitoring Services PIC or an Adaptive Services PIC, you can configure accounting for traffic passing through the routing platform.

To configure discard accounting, include the **accounting group *group-name*** statement at the **[edit forwarding-options]** hierarchy level:

```
[edit forwarding-options]
accounting group-name {
  output {
    cflowd [ hostnames ] {
      aggregation {
        autonomous-system;
        destination-prefix;
        protocol-port;
        source-destination-prefix {
```

```
            caida-compliant;
          }
          source-prefix;
        }
        autonomous-system-type (origin | peer);
        port port-number;
        version format;
      }
      flow-active-timeout seconds;
      flow-inactive-timeout seconds;
      interface {
        engine-id number;
        engine-type number;
        source-address address;
      }
    }
  }
```

To configure the output flow aggregation, include the **cflowd** statement. For more information about flow aggregation, see "Configuring Flow Aggregation (cflowd)" on page 189. To configure the interval before exporting an active flow, include the **flow-active-timeout** statement. The default value for **flow-active-timeout** is **1800** seconds. To configure the interval before a flow is considered inactive, include the **flow-inactive-timeout** statement. The default value for **flow-inactive-timeout** is 60 seconds. To configure the interface that sends out monitored information, include the **interface** statement. Discard accounting is supported for the Monitoring Services PIC only.

When you apply a firewall filter to a loopback interface, the filter might block responses from the Monitoring Services PIC. To allow responses from the Monitoring Services PIC to pass through for accounting purposes, configure a term in the firewall filter to include the Monitoring Services PIC IP address. For more detailed information about configuring firewall filters, see Guidelines for Configuring Standard Firewall Filters and Guidelines for Applying Standard Firewall Filters.

You can use discard accounting for passive and active flow monitoring.

Related Documentation

- *Junos OS Flow Monitoring Feature Guide*
- *Junos OS Class of Service Configuration Guide*

Configuring Flow Monitoring

On routing platforms containing the Monitoring Services PIC or the Monitoring Services II PIC, you can configure flow monitoring for traffic passing through the routing platform. This type of monitoring method is passive monitoring.

To configure flow monitoring, include the **monitoring** statement at the **[edit forwarding-options** hierarchy level:

```
[edit forwarding-options]
monitoring group-name {
  family inet {
```

```
        output {
          cflowd hostname {
            port port-number;
          }
          export-format cflowd-version-5;
          flow-active-timeout seconds;
          flow-export-destination {
            cflowd-collector;
          }
          flow-inactive-timeout seconds;
          interface interface-name {
            engine-id number;
            engine-type number;
            input-interface-index number;
            output-interface-index number;
            source-address address;
          }
        }
      }
    }
```

To configure a passive monitoring group, include the **monitoring** statement and specify a group name. To configure monitoring on a specified address family, include the **family** statement and specify an address family. To specify an interface to monitor incoming traffic, include the **input** statement. To configure the monitoring information that is sent out, include the **output** statement. To configure the output flow aggregation, include the **cflowd** statement. For more information about flow aggregation, see "Configuring Flow Aggregation (cflowd)" on page 189. To specify the format of the monitoring information sent out, include the **export-format** statement and specify a version number. To configure the interval before exporting an active flow, include the **flow-active-timeout** statement. The default value for **flow-active-timeout** is 1800 seconds. To enable flow collection, include the **flow-export-destination** statement. To configure the interval before a flow is considered inactive, include the **flow-inactive-timeout** statement. The default value for **flow-inactive-timeout** is 60 seconds. To configure the interface that sends out the monitored information, include the **interface** statement. Flow monitoring is supported for Monitoring Services PIC interfaces only.

When you apply a firewall filter to a loopback interface, the filter might block responses from the Monitoring Services PIC. To allow responses from the Monitoring Services PIC to pass through for monitoring purposes, configure a term in the firewall filter to include the Monitoring Services PIC's IP address. For more detailed information about configuring firewall filters, see Guidelines for Configuring Standard Firewall Filters and Guidelines for Applying Standard Firewall Filters.

Related Documentation

- *Junos OS Flow Monitoring Feature Guide*
- *Junos OS Class of Service Configuration Guide*

Configuring Load-Balance Groups

In addition to including policers in firewall filters, you can configure a load-balance group that is not part of a firewall filter configuration. A load-balance group contains interfaces that all use the same next-hop group characteristic to load-balance the traffic.

To configure a load-balance group, include the **load-balance-group** statement at the **[edit firewall]** hierarchy level:

```
[edit firewall]
load-balance-group group-name {
  next-hop-group[ group-names ];
}
```

Next-hop groups allow you to include multiple interfaces used to forward duplicate packets used in port mirroring. For more information about next-hop groups, see "Configuring Next-Hop Groups" on page 208.

Configuring Next-Hop Groups

Next-hop groups allow you to include multiple interfaces used to forward duplicate packets used in port mirroring.

To configure a next-hop group, include the **next-hop-group** statement at the **[edit forwarding-options]** hierarchy level:

```
[edit forwarding-options]
next-hop-group [ group-names ] {
  interface interface-name {
    next-hop [ addresses ];
  }
}
```

You can specify one or more group names. To configure the interface that sends out sampled information, include the **interface** statement and specify an interface. To specify a next-hop address to send sampled information, include the **next-hop** statement and specify an IP address.

Next-hop groups have the following restrictions:

- Next-hop groups are supported for M Series routers only.
- Next-hop groups support up to 16 next-hop addresses.
- You can configure up to 30 next-hop groups.
- Each next-hop group must have at least two next-hop addresses.

NOTE: When routes are exported, RIPv2 supports third-party next hops specified in policies, such as Virtual Router Redundancy Protocol (VRRP) groups.

Next-hop groups can be used for port mirroring.

Related Documentation

- Configuring Port Mirroring on page 217

Configuring Per-Prefix Load Balancing

By default, the Junos OS uses a hashing method based only on the destination address to elect a forwarding next hop when multiple equal-cost paths are available. As a result, when multiple routers or switches share the same set of forwarding next hops for a given destination, they can elect the same forwarding next hop.

You can enable router-specific or switch-specific load balancing by including a per-prefix hash value. However, this method applies only to indirect next hops. In other words, when we have a route with a protocol next hop that is not directly connected, it can be resolved over a set of equal-cost forwarding next hops. Only in this case, we use the hashing algorithm to elect a forwarding next hop. An example of this is routes learned from an IBGP neighbor. The protocol next hop for those routes might not be directly reachable and would be resolved through some IGP or static routes. The result could be a set of equal-cost forwarding next hops to reach that protocol next hop. Per-prefix load balancing thus leads to better utilization of the available links.

To configure per-prefix load balancing, include the **load-balance** statement at the **[edit forwarding-options]** hierarchy level:

```
[edit forwarding-options]
load-balance {
  indexed-load-balance;
  per-prefix {
    hash-seed number;
  }
}
```

To enable per-prefix load balancing, you must include the **hash-seed *number*** statement. The range that you can configure is 0 (the default) through 65,535. If no hash seed is configured, the elected forwarding next hop is the same as in previous releases.

If you notice an issue with the load-balance distribution. try including the **indexed-load-balance** statement at the **[edit forwarding-options load-balance]** hierarchy level to see if this resolves the issue. The **indexed-load-balance** statement causes the creation of a nexthop structure that is not a function of the hash only, but is also a function of the low-order bits of the IP address.

CAUTION: Including the **indexed-load-balance** statement causes an increase in memory usage on the device.

```
indexed-load-balance;
```

Configuring Per-Flow Load Balancing Based on Hash Values

By default, the Junos OS uses a hashing method based only on the destination address to elect a forwarding next hop when multiple equal-cost paths are available. All Packet Forwarding Engine slots are assigned the same hash value by default.

You can enable router-specific or switch-specific load balancing by configuring the router or switch to assign a unique, load-balance hash value for each Packet Forwarding Engine slot.

NOTE: **This feature is supported only on M120, M320, and MX Series routers.**

To configure per-flow load balancing. include the **load-balance** statement at the **[edit forwarding-options]** hierarchy level:

```
[edit forwarding-options]
load-balance {
  indexed-load-balance;
  per-flow {
    hash-seed;
  }
}
```

To enable per-flow load balancing, you must include the **hash-seed** statement. The Junos OS automatically chooses a value for the hashing algorithm. You cannot configure a specific value for the **hash-seed** statement when you enable per-flow load balancing.

Configuring Routers, Switches, and Interfaces as DHCP and BOOTP Relay Agents

You can configure the router, switch, or interface to act as a Dynamic Host Configuration Protocol (DHCP) and Bootstrap Protocol (BOOTP) relay agent. This means that a locally attached host can issue a DHCP or BOOTP request as a broadcast message. If the router, switch, or interface sees this broadcast message, it relays the message to a specified DHCP or BOOTP server.

You should configure the router, switch, or interface to be a DHCP and BOOTP relay agent if you have locally attached hosts and a distant DHCP or BOOTP server.

To configure the router or switch to act as a DHCP and BOOTP relay agent, include the **bootp** statement at the **[edit forwarding-options helpers]** hierarchy level:

```
[edit forwarding-options helpers]
bootp {
  client-response-ttl number;
  description text-description;
  interface (interface-name | interface-group) {
    client-response-ttl number;
    description text-description;
    maximum-hop-count number;
    minimum-wait-time seconds;
    no-listen;
```

```
    server address {
      logical-system logical-system-name <routing-instance [ <default>
        routing-instance-names ]>;
      routing-instance [ <default> routing-instance-names ];
    }
  }
  maximum-hop-count number;
  minimum-wait-time seconds;
  relay-agent-option;
  server server-identifier {
    <logical-system logical-system-name>
      <routing-instance [ routing-instance-names ]>;
  }
}
```

To set the description of the BOOTP service, DHCP service, or interface, include the **description** statement.

To set a logical interface or a group of logical interfaces with a specific DHCP relay or BOOTP configuration, include the **interface** statement.

To set the routing instance of the server to forward, include the **routing-instance** statement. You can include as many routing instances as necessary in the same statement.

To stop packets from being forwarded on a logical interface, a group of logical interfaces, or the router or switch, include the **no-listen** statement.

To set the maximum allowed number in the hops field of the BOOTP header, include the **maximum-hop-count** statement. Headers that have a larger number in the hops field are not forwarded. If you omit the **maximum-hop-count** statement, the default value is four hops.

To set the minimum allowed number of seconds in the **secs** field of the BOOTP header, include the **minimum-wait-time** statement. Headers that have a smaller number in the **secs** field are not forwarded. The default value for the minimum wait time is zero (0).

To set the IP address that specify the DHCP or BOOTP server for the router, switch, or interface, include the **server** statement. You can include multiple **server** statements.

To set an IP time-to-live (TTL) value for DHCP response packets sent to a DHCP client, include the **client-response-ttl** statement.

To use the DHCP relay agent option in relayed BOOTP/DHCP messages, include the **relay-agent-option** statement. This option is primarily useful for enabling DHCP forwarding between different VRF routing instances. This option is documented in RFC 3046, *DHCP Relay Agent Information Option*.

You can also configure an individual logical interface to be a DHCP and BOOTP relay agent if you have locally attached hosts and a remote DHCP or BOOTP server connected to one of the router's or switch's interfaces. For more information, see the *Junos OS System Basics Configuration Guide*.

The following example demonstrates a BOOTP relay agent configuration.

```
user@host# show forwarding-options
helpers {
  bootp {
    description "dhcp relay agent global parameters";
    server 192.168.55.44;
    server 172.16.0.3 routing-instance c3;
    maximum-hop-count 10;
    minimum-wait-time 8;
    interface {
      fe-1/3/0 {
        description "use this info for this interface";
        server 10.10.10.10;
        server 192.168.14.14;
        maximum-hop-count 11;
        minimum-wait-time 3;
      }
      fe-1/3/1 {
        no-listen; ###ignore DHCPDISCOVER messages on this interface
      }
      all {
        description "globals apply to all other interfaces";
      }
    }
  }
}
```

Configuring DNS and TFTP Packet Forwarding

You can configure the router or switch to support Domain Name System (DNS) and Trivial File Transfer Protocol (TFTP) packet forwarding for IPv4 traffic, which allows clients to send DNS or TFTP requests to the router or switch. The responding DNS or TFTP server recognizes the client address and sends a response directly to that address. By default, the router or switch ignores DNS and TFTP request packets.

To enable DNS or TFTP packet forwarding, include the **helpers** statement at the **[edit forwarding-options]** hierarchy level:

```
[edit forwarding-options]
helpers {
  domain {
    description text-description;
    interface interface-name {
      description text-description;
      no-listen;
      server [ addresses {
        logical-system logical-system-name;
        routing-instance instance-name;
      }
    }
  }
  tftp {
    description text-description;
    interface interface-name {
      description text-description;
      no-listen;
```

```
        server address;
        server logical-system name < [ routing-instance routing-instance-names ] >;
        server < [ routing-instance routing-instance-names ] >;
      }
    }
  }
```

To set domain packet forwarding, include the **domain** statement.

To set the description of the DNS or TFTP service, include the **description** statement.

To set TFTP packet forwarding, include the **tftp** statement.

To set a DNS or TFTP server (with an IPv4 address), include the **server** statement. Use one address for either a global configuration or for each interface.

To set the routing instance of the server to forward, include the **routing-instance** statement. You can include as many routing instances as necessary in the same statement.

To disable recognition of DNS or TFTP requests on one or more interfaces, include the **no-listen** statement. If you do not specify at least one interface with this statement, the forwarding service is global to all interfaces on the router or switch.

The following sections discuss the following:

- Tracing BOOTP, DNS, and TFTP Forwarding Operations on page 213
- Example: Configuring DNS Packet Forwarding on page 215

Tracing BOOTP, DNS, and TFTP Forwarding Operations

BOOTP, DNS, and TFTP forwarding tracing operations track all BOOTP, DNS, and TFTP operations and record them in a log file. The logged error descriptions provide detailed information to help you solve problems faster.

By default, nothing is traced. If you include the **traceoptions** statement at the **[edit forwarding-options helpers]** hierarchy level, the default tracing behavior is the following:

- Important events are logged in a file called **fud** located in the **/var/log** directory.
- When the file **fud** reaches 128 kilobytes (KB), it is renamed **fud.0**, then **fud.1**, and so on, until there are 3 trace files. Then the oldest trace file (**fud.2**) is overwritten. (For more information about how log files are created, see the *Junos OS System Log Messages Reference*.)
- Log files can be accessed only by the user who configures the tracing operation.

You cannot change the directory (**/var/log**) in which trace files are located. However, you can customize the other trace file settings by including the following statements at the **[edit forwarding-options helpers]** hierarchy level:

```
[edit forwarding-options helpers]
traceoptions {
  file filename <files number> <match regular-expression> <size size> <world-readable |
    no-world-readable>;
  flag {
```

```
        address;
        all;
        config;
        domain;
        ifdb;
        io;
        main;
        port;
        rtsock;
        tftp;
        trace;
        ui;
        util;
    }
    level severity-level;
    no-remote-trace;
}
```

These statements are described in the following sections:

- Configuring the Log Filename on page 214
- Configuring the Number and Size of Log Files on page 214
- Configuring Access to the Log File on page 215
- Configuring a Regular Expression for Lines to Be Logged on page 215

Configuring the Log Filename

By default, the name of the file that records trace output is **fud**. You can specify a different name by including the **file *filename*** statement at the **[edit forwarding-options helpers traceoptions]** hierarchy level:

```
[edit forwarding-options helpers traceoptions]
file filename;
```

Configuring the Number and Size of Log Files

By default, when the trace file reaches 128 kilobytes (KB) in size, it is renamed ***filename*.0**, then ***filename*.1**, and so on, until there are three trace files. Then the oldest trace file (***filename*.2**) is overwritten.

You can configure the limits on the number and size of trace files by including the following statements at the **[edit forwarding-options helpers traceoptions]** hierarchy level:

```
[edit forwarding-options helpers traceoptions]
file files number size size;
```

For example, set the maximum file size to 2 MB, and the maximum number of files to 20. When the file that receives the output of the tracing operation (***filename***) reaches 2 MB, ***filename*** is renamed ***filename*.0**, and a new file called ***filename*** is created. When the new ***filename*** reaches 2 MB, ***filename*.0** is renamed ***filename*.1** and ***filename*** is renamed ***filename*.0**. This process repeats until there are 20 trace files. Then the oldest file (***filename*.19**) is overwritten by the newest file (***filename*.0**).

The number of files can be from 2 through 1000 files. The file size of each file can be from 10 KB through 1 gigabyte (GB).

Configuring Access to the Log File

By default, log files can be accessed only by the user who configures the tracing operation.

To specify that any user can read all log files, include the **world-readable** option with the **file** statement at the **[edit forwarding-options helpers traceoptions]** hierarchy level:

```
[edit forwarding-options helpers traceoptions]
file world-readable;
```

To explicitly set the default behavior, include the **no-world-readable** option with the **file** statement at the **[edit forwarding-options helpers traceoptions]** hierarchy level:

```
[edit forwarding-options helpers traceoptions]
file no-world-readable;
```

Configuring a Regular Expression for Lines to Be Logged

By default, the trace operation output includes all lines relevant to the logged events.

You can refine the output by including the **match** option with the **file** statement at the **[edit forwarding-options helpers traceoptions]** hierarchy level and specifying a regular expression (regex) to be matched:

```
[edit forwarding-options helpers traceoptions]
file filename match regular-expression;
```

Example: Configuring DNS Packet Forwarding

Enable DNS packet request forwarding to all interfaces on a router except **t1-1/1/2** and **t1-1/1/3**:

```
[edit forwarding-options helpers]
dns {
  server 10.10.10.30;
  interface {
    t1-1/1/2 {
      no-listen;
      server 10.10.10.9;
    }
    t1-1/1/3 {
      no-listen;
      server 10.10.10.4;
    }
  }
}
```

Preventing DHCP Spoofing on MX Series 3D Universal Edge Routers

A problem that sometimes occurs with DHCP is *DHCP spoofing*. in which an untrusted client floods a network with DHCP messages. Often these attacks utilize source IP address spoofing to conceal the true source of the attack.

DHCP snooping helps prevent DHCP spoofing by copying DHCP messages to the control plane and using the information in the packets to create anti-spoofing filters. The anti-spoofing filters bind a client's MAC address to its DHCP-assigned IP address and use this information to filter spoofed DHCP messages. In a typical topology, a carrier edge router (in this function also referred to as the broadband services router [BSR]) connects the DHCP server and the MX Series router (or broadband services aggregator [BSA]) performing the snooping. The MX Series router connects to the client and the BSR.

DHCP snooping works as follows in the network topology mentioned above:

1. The client sends a DHCP discover message to obtain an IP address from the DHCP server.
2. The BSA intercepts the message and might add option 82 information specifying the slot, port, VPI/VCI, and so on.
3. The BSA then sends the DHCP discover message to the BSR, which converts it to a unicast packet and sends it to the DHCP server.
4. The DHCP server looks up the client's MAC address and option 82 information in its database. A valid client is assigned an IP address, which is returned to the client using a DHCP offer message. Both the BSR and BSA send this message upstream to the client.
5. The client examines the DHCP offer, and if it is acceptable, issues a DHCP request message that is sent to the DHCP server through the BSA and BSR.
6. The DHCP server confirms that the IP address is still available. If it is, the DHCP server updates its local tables and sends a DHCP ACK message to the client.
7. The BSR receives the DHCP ACK message and passes the message to the BSA.
8. The BSA creates an anti-spoofing filter by binding the IP address in the ACK message to the MAC address of the client. After this point, any DHCP messages from this IP address that are not bound to the client's MAC address are dropped.
9. The BSA sends the ACK message to the client so that the process of assigning a IP address can be completed.

You configure DHCP snooping by including within a DHCP group the appropriate interfaces of the BSA:

```
[edit routing-instances routing-instance-name bridge-domains bridge-domain-name
    forwarding-options dhcp-relay group group-name]
interface interface-name;
```

In a VPLS environment, DHCP requests are forwarded over pseudowires. You can configure DHCP snooping over VPLS at the **[edit routing-instances *routing-instance-name*]** hierarchy level.

DHCP snooping works on a per learning bridge basis in bridge domains. Each learning domain must have an upstream interface configured. This interface acts as the flood port for DHCP requests coming from the client side. DHCP requests are be forwarded across learning domains in a bridge domain. You can configure DHCP snooping on bridge

domains at the **[edit routing-instances *routing-instance-name* bridge-domains *bridge-domain-name*]** hierarchy level.

Related Documentation

- *Junos OS MX Series 3D Universal Edge Routers Solutions Guide*

Configuring Port Mirroring

Port mirroring is the ability of a router to send a copy of an IPv4 or IPv6 packet to an external host address or a packet analyzer for analysis. Port mirroring is different from traffic sampling. In traffic sampling, a sampling key based on the packet header is sent to the Routing Engine. There, the key can be placed in a file, or cflowd packets based on the key can be sent to a cflowd server. In port mirroring, the entire packet is copied and sent out through a next-hop interface.

One application for port mirroring sends a duplicate packet to a virtual tunnel. A next-hop group can then be configured to forward copies of this duplicate packet to several interfaces. For more information about next-hop groups, see "Configuring Next-Hop Groups" on page 208.

All M Series Multiservice Edge Routers, T Series Core Routers, and MX Series 3D Universal Edge Routers support port mirroring for IPv4 or IPv6. The M120, M320, and MX Series routers support port mirroring for IPv4 and IPv6 simultaneously.

Port mirroring for VPLS traffic is supported on M7i and M10i routers configured with an Enhanced CFEB (CFEB-E), on M120 routers, on M320 routers configured with an Enhanced III Flexible PIC Concentrators (FPCs), and MX Series routers.

In Junos OS Release 9.3 and later, port mirroring is supported for Layer 2 traffic on MX Series routers. For information about how to configure port mirroring for Layer 2 traffic, see the *Junos OS Layer 2 Configuration Guide*.

In Junos OS Release 9.6 and later, port mirroring is supported for Layer 2 VPN traffic on M120 routers and M320 routers configured with an Enhanced III FPC. You can also set the maximum length of the mirrored packet. When set, the mirrored packet is truncated to the specified length.

Configuration Guidelines

When configuring port mirroring, the following restrictions apply:

- Only transit data is supported.
- You can configure either IPv4 or IPv6 port mirroring but not both on M Series routers, except for the M120 and M320 routers, which support port mirroring for IPv4 and IPv6 simultaneously.
- You can configure port mirroring for IPv4 and IPv6 simultaneously on the M120 and M320 routers and the MX Series routers.
- Ingress filtering of multicast packets is supported on all Dense Port Concentrators (DPCs) in MX Series routers. Egress filtering of multicast packets is supported for interfaces on Trio-based DPCs in MX Series routers only. Filtering of multicast packets

based on destination address is not supported on M Series routers or T Series routers and is not supported for interfaces on I-chip ASIC-based DPCs in MX Series routers.

- By default, firewall filters cannot be applied to port-mirroring destination interfaces. To enable port-mirroring destination interfaces to support firewall filters, use the **no-filter-check** statement to disable filter checking on the interfaces. You can include the **no-filter-check** statement at the following hierarchy levels:
  - **[edit forwarding-options port-mirroring family (inet | inet6 | ccc | vpls) output]**
  - **[edit forwarding-options port-mirroring instance *instance-name* family (inet | ccc | vpls) output]**
- You must include a firewall filter with both the **accept** action and the **port-mirror** action modifier on the inbound interface. Port mirroring does not work if you specify the **discard** action.
- The interface you configure for port mirroring should not participate in any kind of routing activity.
- The destination address you specify should not have a route to the ultimate traffic destination. For example, if the sampled IPv4 packets have a destination address of **192.68.9.10** and the port-mirrored traffic is sent to **192.68.20.15** for analysis, the device associated with the latter address should not know a route to **192.68.9.10**. Also, it should not send the sampled packets back to the source address.
- On all routers except the MX Series router, you can configure only one port-mirroring interface per router. If you include more than one interface in the **port-mirroring** statement, the previous one is overwritten. MX Series routers support more than one port-mirroring interface per router.
- You can configure multiple port-mirroring instances on the M120, M320, and MX Series routers.
- You can specify both host (cflowd) sampling and port mirroring in the same configuration. You can perform RE-sampling and port mirroring actions simultaneously. However, you cannot perform PIC-sampling and port mirroring actions simultaneously.
- In typical applications, you send the sampled packets to an analyzer or a workstation for analysis, not to another router. If you must send this traffic over a network, you should use tunnels.

Configuring Port Mirroring

To configure port mirroring, include the **port-mirroring** statement at the **[edit forwarding-options]** hierarchy level:

```
[edit forwarding-options]
port-mirroring {
  family (ccc | inet | inet6 | vpls) {
    output {
      interface interface-name {
        next-hop address;
      }
      no-filter-check;
```

```
    }
    input {
      maximum-packet-length bytes;
      rate number;
      run-length number;
    }
  }
}
```

Configuring the Port-Mirroring Address Family and Interface

To configure port mirroring, include the **port-mirroring** statement. To configure the address family type of traffic to sample, include the **family** statement. To configure the rate of sampling, length of sampling, and the maximum size for the mirrored packet, include the **input** statement. To specify on which interface to send duplicate packets and the next-hop address to send packets, include the **output** statement. To determine whether there are any filters on the specified interface, include the **no-filter-check** statement.

For information about the **rate** and **run-length** statements, see "Configuring Traffic Sampling" on page 185.

Configuring Multiple Port-Mirroring Instances

In Junos OS Release 9.5 and later, you can configure multiple port-mirroring instances on the M120, M320, and MX Series routers. On the M120 router, you can associate each instance with a specific Forwarding Engine Board (FEB). You cannot associate a port-mirroring instance with an FEB configured as a backup FEB. On the M320 router, you can associate each instance with a specific Flexible PIC Concentrator (FPC). Associating a port-mirroring instance with an FPC or an FEB enables you to mirror packets to different destinations. Multiple port-mirroring instances are also supported on MX Series routers. For information about configuring multiple port-mirroring instances on MX Series routers, see the *Junos OS Layer 2 Configuration Guide*.

To configure a port-mirroring instance, include the **instance** *port-mirroring-instance* statement at the **[edit forwarding-options port-mirroring]** hierarchy level:

```
[edit forwarding-options port-mirroring]
instance port-mirroring-instance-name {
  family (ccc | inet | inet6 | vpls) {
    output {
      interface interface-name {
        next-hop address;
      }
      no-filter-check;
    }
  }
  input {
    maximum-packet-length bytes;
    rate number;
    run-length number;
  }
}
```

Configuring Port-Mirroring Instances

You can configure multiple port-mirroring instances. Specify a unique ***port-mirroring-instance-name*** for each instance you configure.

Associating a Port-Mirroring Instance on M320 Routers

You can associate a port-mirroring instance with a specific FPC on an M320 router or with a specific FEB on an M120 router. You can associate only one port-mirroring instance with each FPC on an M320 router or with each FEB on an M120 router. On an M120 router, you cannot associate a port-mirroring instance with a FEB configured as a backup FEB.

To associate a port-mirroring instance with an FPC on an M320 router, include the **port-mirror-instance *port-mirroring-instance-name*** statement at the **[edit chassis fpc *slot-number*]** hierarchy level:

```
[edit chassis]
fpc slot-number {
    port-mirror-instance port-mirroring-instance-name;
}
```

For ***slot-number***, specify the slot number of the FPC you want to associate with the port-mirroring instance. For ***port-mirroring-instance-name***, specify the name of a port-mirroring instance you configured at the **[edit forwarding-options port-mirroring]** hierarchy level. For more information about configuring an FPC on an M320 router, see the *Junos OS System Basics Configuration Guide*.

Associating a Port-Mirroring Instance on M120 Routers

To associate a port-mirroring instance with a FEB on an M120 router, include the **port-mirror-instance *port-mirroring-instance-name*** statement at the **[edit chassis feb *slot-number*]** hierarchy level:

```
[edit chassis]
feb slot-number {
    port-mirror-instance port-mirroring-instance-name;
}
```

For ***slot-number***, specify the slot number of the FEB you want to associate with the port-mirroring instance. For ***port-mirroring-instance-name***, specify the name of a port-mirroring instance you configured at the **[edit forwarding-options port-mirroring]** hierarchy level. For information about configuring FEB redundancy on an M120 router, see the *Junos OS High Availability Configuration Guide*. For information about configuring FPC-to-FEB connectivity on an M120 router, see the *Junos OS System Basics Configuration Guide*.

Configuring MX Series 3D Universal Edge Routers and M120 Routers to Mirror Traffic Only Once

On MX Series and M120 routers only, you can configure port mirroring so that the router mirrors traffic only once. If you configure port mirroring on both ingress and egress interfaces, the same packet could be mirrored twice. To mirror packets only once and prevent the router from sending duplicate sampled packets to the same mirroring

destination, include the **mirror-once** statement at the **[edit forwarding-options port-mirroring]** hierarchy level:

```
[edit forwarding-options port-mirroring]
mirror-once;
```

NOTE: The **mirror-once** statement is supported only in the global port-mirroring instance.

Configuring Packet Capture

Packet capture allows you to monitor and analyze offline IP version 4 (IPv4) packets flowing through a router. Packet capture monitors packet fragments also. Packet capture can be enabled on any interface and can analyze ingress traffic, egress traffic, or both.

NOTE: Packet capture is supported for the J Series Services Routers only. Packet capture is not supported on tunnel interfaces. You cannot configure packet capture and sampling at the same time.

To configure packet capture, include the **packet-capture** statement at the **[edit forwarding-options]** hierarchy level:

```
[edit forwarding-options]
packet-capture {
  disable;
  file filename file-name <files number> <size number> <world-readable |
    no-world-readable>;
  maximum-capture-size bytes;
}
```

To disable packet capture, include the **disable** statement. Packet capture is enabled by default.

You can capture packets into files. Files are classified based on the physical interface the packets are captured on (one file per physical interface). You can specify the file name, maximum size, and maximum number of files. When you capture a file named **pcap-file**, packet capture creates one file for each physical interface and appends the physical interface designator to the filename (for example, **at**). When the file named **pcap-file.*xx*** reaches its maximum size, the file is renamed **pcap-file.*xx*.0**. When **pcap-file.*xx*** reaches its maximum size again, the file is renamed **pcap-file.*xx*.1**. This process continues until the maximum number of files is exceeded. When that happens, the oldest file is overwritten. The file named **pcap-file.*xx*** is always the latest file. The packet capture file for an interface is created when the first packet is captured on that interface. Once created, this file is not removed even if packet capture is disabled on the interface. All packet capture files are stored in the **/var/tmp/** directory.

If the PCAP file is deleted from the var/tmp/ directory, the file is not re-created upon the next packet capture traffic on the interface. You must first disable and then enable PCAP functionality again to re-create the PCAP file.

To enable capture into files, include the **file** statement. You can specify the target filename, maximum file size, and the maximum number of files. To specify the name of the target file, include the **filename** statement. To specify the maximum size of the file, include the **size** statement. To specify the maximum number of files, include the **files** statement.

To specify the maximum size of the packet for capture, include the **maximum-capture-size** statement.

You can capture packets on a specific interface by configuring either of the following:

- Configure a firewall filter with the action **sample** and apply it to the interface.
- Configure sampling on the interface in the ingress or egress traffic.

NOTE: Interface sampling does not capture host-originated packets. Configure firewall filters to capture host-originated packets.

NOTE: A firewall filter applied to a loopback interface (lo0) affects all packets going to and from the Routing Engine.

You can capture packets on a specific interface.

You can capture only specific types of packets by using a firewall filter in conjunction with packet capture. To configure packet capture for specific packets using firewall filters, include the following statements at the **[edit firewall]** hierarchy level:

```
[edit firewall]
filter filter-name {
   term term-name {
      from {
         match-conditions;
      }
      then {
         sample;
         accept;
      }
   }
}
```

NOTE: Configure packet capture with appropriate firewall filters to control the number of packets captured. Performance of the router may be impacted if packet capture is used without configuring any firewall filters.

NOTE: Packet capture does not support multilink encapsulations (such as MLPPP).

You must disable packet capture to modify encapsulation. To modify the encapsulation on a packet capture-enabled interface, perform the following tasks:

1. Disable packet capture by including the **disable** statement at the **[edit forwarding-options packet-capture]** hierarchy level.
2. Remove the packet capture file for the interface from the **/var/tmp/** directory.
3. Change the encapsulation.
4. Enable packet capture.

For packets captured on T1, T3, E1, E3, SE, and ISDN interfaces in the egress direction, the size of packets captured can be one byte less than the configured value of **maximum-capture-size** because of the PLP byte.

To capture packets on an ISDN interface, configure packet capture on the dialer interface. To capture packets on the PPPoE interface, configure packet capture on the PPPoE interface.

Packet capture is not supported with MLPPP encapsulation. However, the CLI does not prevent you from enabling packet capture on an interface with MLPPP encapsulation. If packet capture is enabled in the input direction on an interface with MLPPP encapsulation, input packets on that interface are captured on the output interfaces.

By default, there is no tracing operation support for packet capture.

PART 3

Administration

CHAPTER 13

Routing Policy Reference

Protocols That Can Be Imported to and Exported from the Routing Table

Table 20: Protocols That Can Be Imported to and Exported from the Routing Table

Protocol	Import	Export
BGP	Yes	Yes
Distance Vector Multicast Routing Protocol (DVMRP)	Yes	Yes
IS-IS	Yes	Yes
LDP	Yes	Yes
MPLS	Yes	No
OSPF	Yes	Yes
Protocol Independent Multicast (PIM) dense mode	Yes	Yes
PIM sparse mode	Yes	Yes
PIM sparse-dense mode	Yes	Yes

Table 20: Protocols That Can Be Imported to and Exported from the Routing Table *(continued)*

Protocol	Import	Export

Pseudoprotocol:
• Direct routes
• Explicitly configured routes
  • Aggregate routes
  • Generated routes
  • Local routes
  • Static routes
	Yes	No

Routing Information Protocol (RIP) and Routing Information Protocol next generation (RIPng)	Yes	Yes

Routing Tables Affected by Routing Policies

Table 21 on page 228 lists the routing tables affected by default and user-defined routing policies and the types of routes that each routing table stores.

Table 21: Routing Tables Affected by Routing Policies

Routing Table	Type of Routes Stored
inet.0	Unicast IPv4 routes
.inet.0 ***instance-name***	Unicast IPv4 routes for a particular routing instance
inet.1	Multicast IPv4 routes
inet.2	Unicast IPv4 routes for multicast reverse-path forwarding (RPF) lookup
inet.3	MPLS routes
inet.flow.0	Flow-specification routes
instance-name**.inetflow.0**	Flow-specification routes for Layer 3 VPNs
mpls.0	MPLS routes for label-switched path (LSP) next hops
inet6.0	Unicast IP version 6 (IPv6) routes

NOTE: The discussion in the rest of this chapter assumes that the routing table is inet.0 unless explicitly stated otherwise.

Default Import and Export Policies for Protocols

Table 22: Default Import and Export Policies for Protocols

Importing or Exporting Protocol	Default Import Policy	Default Export Policy
BGP	Accept all received BGP IPv4 routes learned from configured neighbors and import into the **inet.0** routing table. Accept all received BGP IPv6 routes learned from configured neighbors and import into the **inet6.0** routing table.	Readvertise all learned BGP routes to all BGP speakers, while following protocol-specific rules that prohibit one IBGP speaker from readvertising routes learned from another IBGP speaker, unless it is functioning as a route reflector.
DVMRP	Accept all DVMRP routes and import into the **inet.1** routing table.	Accept and export active DVMRP routes.
IS-IS	Accept all IS-IS routes and import into the **inet.0** and **inet6.0** routing tables. (You cannot override or change this default policy.)	Reject everything. (The protocol uses flooding to announce local routes and any learned routes.)
LDP	Accept all LDP routes and import into the **inet.3** routing table.	Reject everything.
MPLS	Accept all MPLS routes and import into the **inet.3** routing table.	Accept and export active MPLS routes.
OSPF	Accept all OSPF routes and import into the **inet.0** routing table. (You cannot override or change this default policy.)	Reject everything. (The protocol uses flooding to announce local routes and any learned routes.)
PIM dense mode	Accept all PIM dense mode routes and import into the **inet.1** routing table.	Accept active PIM dense mode routes.
PIM sparse mode	Accept all PIM sparse mode routes and import into the **inet.1** routing table.	Accept and export active PIM sparse mode routes.

Pseudoprotocol:
• Direct routes
• Explicitly configured routes:
  • Aggregate routes
  • Generated routes
  • Static routes
	Accept all direct and explicitly configured routes and import into the **inet.0** routing table.	The pseudoprotocol cannot export any routes from the routing table because it is not a routing protocol.
		
		Routing protocols can export these or any routes from the routing table.

Table 22: Default Import and Export Policies for Protocols *(continued)*

Importing or Exporting Protocol	Default Import Policy	Default Export Policy
RIP	Accept all RIP routes learned from configured neighbors and import into the **inet.0** routing table.	Reject everything. To export RIP routes, you must configure an export policy for RIP.
RIPng	Accept all RIPng routes learned from configured neighbors and import into the **inet6.0** routing table.	Reject everything. To export RIPng routes, you must configure an export policy for RIPng.
Test policy	Accept all routes. For additional information about test policy, see "Routing Policy Tests" on page 26.	

Routing Policy Match Conditions

A *match condition* defines the criteria that a route must match. You can define one or more match conditions. If a route matches all match conditions, one or more actions are applied to the route.

Match conditions fall into two categories: standard and extended. In general, the extended match conditions include criteria that are defined separately from the routing policy (AS path regular expressions, communities, and prefix lists) and are more complex than standard match conditions. The extended match conditions provide many powerful capabilities. The standard match conditions include criteria that are defined within a routing policy and are less complex than the extended match conditions, also called named match conditions.

Table 23 on page 230 describes each match condition, including its category, when you typically use it, and any relevant notes about it. For more information about match conditions, see "Configuring Match Conditions in Routing Policy Terms" on page 47.

Table 23: Match Conditions

Match Condition	Category	When to Use	Notes
AS path regular expression—A combination of AS numbers and regular expression operators.	Extended	(BGP only) Match a route based on its AS path. (An AS path consists of the AS numbers of all routers a packet must go through to reach a destination.) You can specify an exact match with a particular AS path or a less precise match.	You use regular expressions to match the AS path.

Table 23: Match Conditions *(continued)*

Match Condition	Category	When to Use	Notes
Community—A group of destinations that share a property. (Community information is included as a path attribute in BGP update messages.)	Extended	Match a group of destinations that share a property. Use a routing policy to define a community that specifies a group of destinations you want to match and one or more actions that you want taken on this community.	Actions can be performed on the entire group. You can create multiple communities associated with a particular destination. You can create match conditions using regular expressions.
Prefix list—A named list of IP addresses.	Extended	Match a route based on prefix information. You can specify an exact match of a particular route only.	You can specify a common action only for all prefixes in the list.
Route list—A list of destination prefixes.	Extended	Match a route based on prefix information. You can specify an exact match of a particular route or a less precise match.	You can specify an action for each prefix in the route list or a common action for all prefixes in the route list.
Standard—A collection of criteria that can match a route.	Standard	Match a route based on one of the following criteria: area ID, color, external route, family, instance (routing), interface name, level number, local preference, metric, neighbor address, next-hop address, origin, preference, protocol, routing table name, or tag. For the protocol criterion, you can specify one of the following: BGP, direct, DVMRP, IS-IS, local, MPLS, OSPF, PIM dense mode, PIM sparse mode, RIP, RIPng, static, and aggregate.	None.
Subroutine—A routing policy that is called repeatedly from another routing policy.	Extended	Use an effective routing policy in other routing policies. You can create a subroutine that you can call over and over from other routing policies.	The subroutine action influences but does not necessarily determine the final action. For more information, see "How a Routing Policy Subroutine Is Evaluated" on page 24.

Protocol Support for Import and Export Policies

Table 24: Protocol Support for Import and Export Policies

Protocol	Import Policy	Export Policy	Supported Levels
BGP	Yes	Yes	Import: global, group, peer Export: global, group, peer
DVMRP	Yes	Yes	Global

Table 24: Protocol Support for Import and Export Policies *(continued)*

Protocol	Import Policy	Export Policy	Supported Levels
IS-IS	No	Yes	Export: global
LDP	Yes	Yes	Global
MPLS	No	No	–
OSPF	Yes	Yes	Export: global Import: external routes only
PIM dense mode	Yes	Yes	Global
PIM sparse mode	Yes	Yes	Global
Pseudoprotocol—Explicitly configured routes, which include the following: • Aggregate routes • Generated routes	Yes	No	Import: global
RIP and RIPng	Yes	Yes	Import: global, neighbor Export: group

CHAPTER 14

Summary of Routing Policy Configuration Statements

aigp-originate

Syntax aigp-originate *distance*;

Hierarchy Level [edit logical-systems *logical-system-name* policy-options policy-statement *policy-name* term *term-name* then],
[edit logical-systems *logical-system-name* policy-options policy-statement *policy-name* then],
[edit policy-options policy-statement *policy-name* term *term-name* then],
[edit policy-options policy-statement *policy-name* then]

Release Information Statement introduced in Junos OS Release 12.1.

Description Originate an accumulated interior gateway protocol (AIGP) BGP attribute for a given prefix by export policy, using the **aigp-originate** policy action.

To originate an AIGP attribure, you need configure the policy action on only one node. The AIGP attribute is readvertised if the neighbors are AIGP enabled with the **aigp** statement in the BGP configuration.

Default If you omit the **aigp-originate** policy action, the node still readvertises the AIGP BGP attribute if AIGP is enabled in the BGP configuration. However, the node does not originate its own AIGP attribute for local prefixes.

As the route is readvertised by downstream nodes, the cost of the AIGP attribute reflects the IGP distance to the prefix (zero + IGP distance or configured distance + IGP distance).

Options ***distance***—(Optional) Associate an initial cost when advertising a local prefix with the AIGP BGP attribute.

Range: 0 through 4,294,967,295

Default: The initial cost associated with the AIGP attribute for a local prefix is zero. The ***distance*** option overrides the default zero value for the initial cost.

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

Related Documentation

- Example: Configuring the Accumulated IGP Attribute for BGP
- aigp

apply-path

Syntax apply-path *path*;

Hierarchy Level [edit logical-systems *logical-system-name* policy-options prefix-list *name*],
[edit policy-options prefix-list *name*]

Release Information Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Expand a prefix list to include all prefixes pointed to by a defined path.

Options *path*—String of elements composed of identifiers or configuration keywords that points to a set of prefixes. You can include wildcards (enclosed in angle brackets) to match more than one identifier. You cannot add a path element, including wildcards, after a leaf statement. Path elements, including wildcards, can only be used after a container statement.

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

Related Documentation
- Configuring Prefix Lists on page 128

as-path

Syntax as-path *name regular-expression*;

Hierarchy Level [edit dynamic policy-options],
[edit logical-systems *logical-system-name* policy-options],
[edit policy-options]

Release Information Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.
Support for configuration in the dynamic database introduced in Junos OS Release 9.5.
Support for configuration in the dynamic database introduced in Junos OS Release 9.5 for EX Series switches.

Description Define an autonomous system (AS) path regular expression for use in a routing policy match condition.

Options ***name***—Name that identifies the regular expression. The name can contain letters, numbers, and hyphens (-) and can be up to 65,536 characters long. To include spaces in the name, enclose it in quotation marks (“ ”).

regular-expression—One or more regular expressions used to match the AS path.

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

Related Documentation

- Configuring AS Path Regular Expressions to Use as Routing Policy Match Conditions on page 109
- Configuring Routing Policies and Policy Objects in the Dynamic Database on page 71
- dynamic-db on page 242

as-path-group

Syntax

```
as-path-group group-name {
  as-path name regular-expression;
}
```

Hierarchy Level

[edit dynamic policy-options],
[edit logical-systems *logical-system-name* policy-options],
[edit policy-options]

Release Information

Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.
Support for dynamic database configuration introduced in Junos OS Release 9.5.
Support for dynamic database configuration introduced in Junos OS Release 9.5 for EX Series switches.

Description

Define a group containing multiple AS path regular expressions for use in a routing policy match condition.

Options

group-name—Name that identifies the AS path group. One or more AS path regular expressions must be listed below the **as-path-group** hierarchy.

name—Name that identifies the regular expression. The name can contain letters, numbers, and hyphens (-) and can be up to 255 characters long. To include spaces in the name, enclose it in quotation marks (“ ”).

regular-expression—One or more regular expressions used to match the AS path.

Required Privilege Level

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

Related Documentation

- Configuring AS Path Regular Expressions to Use as Routing Policy Match Conditions on page 109
- Configuring Routing Policies and Policy Objects in the Dynamic Database on page 71
- dynamic-db on page 242

community

Syntax

```
community name {
  invert-match;
  members [ community-ids ];
}
```

Hierarchy Level

[edit dynamic policy-options],
[edit logical-systems *logical-system-name* policy-options],
[edit policy-options]

Release Information

Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.
Support for configuration in the dynamic database introduced in Junos OS Release 9.5.
Support for configuration in the dynamic database introduced in Junos OS Release 9.5 for EX Series switches.

Description

Define a community or extended community for use in a routing policy match condition.

Options

name—Name that identifies the regular expression. The name can contain letters, numbers, and hyphens (-) and can be up to 255 characters. To include spaces in the name, enclose it in quotation marks (“ ”).

invert-match—Invert the results of the community expression matching.

members *community-ids*—One or more community members. If you specify more than one member, you must enclose all members in brackets.

The format for ***community-ids*** is:

as-number:*community-value*

as-number is the AS number and can be a value in the range from 0 through 65,535. ***community-value*** is the community identifier and can be a number in the range from 0 through 65,535.

You also can specify ***community-ids*** for communities as one of the following well-known community names, which are defined in RFC 1997, *BGP Communities Attribute*:

- **no-export**—Routes containing this community name are not advertised outside a BGP confederation boundary.
- **no-advertise**—Routes containing this community name are not advertised to other BGP peers.
- **no-export-subconfed**—Routes containing this community name are not advertised to external BGP peers, including peers in other members' ASs inside a BGP confederation.

You can explicitly exclude BGP community information with a static route using the **none** option. Include **none** when configuring an individual route in the **route** portion of the **static** statement to override a **community** option specified in the **defaults** portion of the statement.

The format for extended ***community-ids*** is the following:

type:*administrator*:*assigned-number*

type is the type of extended community and can be either a **bandwidth**, **target**, **origin**, **domain-id**, **src-as**, or **rt-import** community or a 16-bit number that identifies a specific BGP extended community. The **target** community identifies the destination to which the route is going. The **origin** community identifies where the route originated. The **domain-id** community identifies the OSPF domain from which the route originated. The **src-as** community identifies the autonomous system from which the route originated. The **rt-import** community identifies the route to install in the routing table.

NOTE: For **src-as**, you can specify only an AS number and not an IP address. For **rt-import**, you can specify only an IP address and not an AS number.

administrator is the administrator. It is either an AS number or an IPv4 address prefix, depending on the type of extended community.

assigned-number identifies the local provider.

The format for linking a bandwidth with an AS number is:

bandwidth:*as-number*:*bandwidth*

as-number specifies the AS number and ***bandwidth*** specifies the bandwidth in bytes per second.

NOTE: In Junos OS Release 9.1 and later, you can specify 4-byte AS numbers as defined in RFC 4893, *BGP Support for Four-octet AS Number Space*, as well as the 2-byte AS numbers that are supported in earlier releases of the Junos OS. In plain-number format, you can configure a value in the range from 1 through 4,294,967,295. To configure a **target** or **origin** extended community that includes a 4-byte AS number in the plain-number format, append the letter "L" to the end of number. For example, a target community with the 4-byte AS number 334,324 and an assigned number of 132 is represented as **target:334324L:132**.

In Junos OS Release 9.2 and later, you can also use AS-dot notation when defining a 4-byte AS number for the **target** and **origin** extended communities. Specify two integers joined by a period: *16-bit high-order value in decimal*.*16-bit low-order value in decimal*. For example, the 4-byte AS number represented in plain-number format as 65546 is represented in AS-dot notation as 1.10.

Required Privilege Level

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

Related Documentation

- Overview of BGP Communities and Extended Communities as Routing Policy Match Conditions on page 27
- Defining BGP Communities and Extended Communities for Use in Routing Policy Match Conditions on page 116
- Configuring Routing Policies and Policy Objects in the Dynamic Database on page 71
- dynamic-db on page 242

condition

Syntax

```
condition condition-name {
    if-route-exists address table table-name;
}
```

Hierarchy Level

[edit dynamic policy-options],
[edit logical-systems *logical-system-name* policy-options],
[edit policy-options]

Release Information

Statement introduced in Junos OS Release 9.0.
Statement introduced in Junos OS Release 9.0 for EX Series switches.
Support for configuration in the dynamic database introduced in Junos OS Release 9.5.
Support for configuration in the dynamic database introduced in Junos OS Release 9.5 for EX Series switches.

Description

Define a policy condition based on the existence of routes in specific tables for use in BGP export policies.

Options

if-route-exists *address*—Specify the address of the route in question.

table *table-name*—Specify a routing table.

Required Privilege Level

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

Related Documentation

- Configuring Routing Policy Match Conditions Based on Routing Table Entries on page 150
- Configuring Routing Policies and Policy Objects in the Dynamic Database on page 71
- dynamic-db on page 242

damping

Syntax

```
damping name {
  disable;
  half-life minutes;
  max-suppress minutes;
  reuse number;
  suppress number;
}
```

Hierarchy Level [edit logical-systems *logical-system-name* policy-options],
[edit policy-options]

Release Information Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Define route flap damping properties to set on BGP routes.

Options

disable—Disable damping on a per-prefix basis. Any damping state that is present in the routing table for a prefix is deleted if damping is disabled.

half-life *minutes*—Decay half-life. ***minutes*** is the interval after which the accumulated figure-of-merit value is reduced by half if the route remains stable.

Range: 1 through 45

Default: 15 minutes

NOTE: **For the half-life, configure a value that is less than the max-suppress. If you do not, the configuration is rejected.**

max-suppress *minutes*—Maximum hold-down time. ***minutes*** is the maximum time that a route can be suppressed no matter how unstable it has been.

Range: 1 through 720

Default: 60 minutes

NOTE: **For the max-suppress, configure a value that is greater than the half-life. If you do not, the configuration is rejected.**

name—Name that identifies the set of damping parameters. The name can contain letters, numbers, and hyphens (-) and can be up to 255 characters long. To include spaces in the name, enclose it in quotation marks (“ ”).

reuse *number*—Reuse threshold. ***number*** is the figure-of-merit value below which a suppressed route can be used again.

Range: 1 through 20,000

Default: 750 (unitless)

suppress *number*—Cutoff (suppression) threshold. ***number*** is the figure-of-merit value above which a route is suppressed for use or inclusion in advertisements.

Range: 1 through 20,000

Default: 3000 (unitless)

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

Related Documentation
- Configuring BGP Flap Damping Parameters on page 155

dynamic-db

Syntax
dynamic-db;

Hierarchy Level
[edit logical-systems *logical-system-name* policy-options as-path *path-name*],
[edit logical-systems *logical-system-name* policy-options as-path-group *group-name*],
[edit logical-systems *logical-system-name* policy-options community *community-name*],
[edit logical-systems *logical-system-name* policy-options condition *condition-name*],
[edit logical-systems *logical-system-name* policy-options policy-statement *policy-statement-name*],
[edit logical-systems *logical-system-name* policy-options prefix-list *prefix-list-name*],
[edit policy-options as-path *path-name*],
[edit policy-options as-path-group *group-name*],
[edit policy-options community *community-name*],
[edit policy-options condition *condition-name*],
[edit policy-options policy-statement *policy-statement-name*],
[edit policy-options prefix-list *prefix-list-name*]

Release Information
Statement introduced in Junos OS Release 9.5.
Statement introduced in Junos OS Release 9.5 for EX Series switches.

Description
Define routing policies and policy objects that reference policies configured in the dynamic database at the **[edit dynamic]** hierarchy level.

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

Related Documentation
- Configuring Routing Policies Based on Dynamic Database Configuration on page 72

export

Syntax export [*policy-names*];

Hierarchy Level
[edit logical-systems *logical-system-name* protocols bgp],
[edit logical-systems *logical-system-name* protocols bgp group *group-name*],
[edit logical-systems *logical-system-name* protocols bgp group *group-name* neighbor *address*],
[edit logical-systems *logical-system-name* protocols bgp group *group-name* neighbor *address* out-delay *seconds*],
[edit logical-systems *logical-system-name* protocols dvmrp],
[edit logical-systems *logical-system-name* protocols isis],
[edit logical-systems *logical-system-name* protocols ldp],
[edit logical-systems *logical-system-name* protocols msdp],
[edit logical-systems *logical-system-name* protocols msdp group *group-name*],
[edit logical-systems *logical-system-name* protocols msdp group *group-name* peer *address*],
[edit logical-systems *logical-system-name* protocols ospf],
[edit logical-systems *logical-system-name* protocols ospf3],
[edit logical-systems *logical-system-name* protocols pim rp bootstrap family (inet | inet6)],
[edit logical-systems *logical-system-name* rip group *group-name*],
[edit logical-systems *logical-system-name* ripng group *group-name*],
[edit protocols bgp],
[edit protocols bgp group *group-name*],
[edit protocols bgp *group-name* neighbor *address*],
[edit protocols bgp group *group-name* neighbor *address* out-delay *seconds*],
[edit protocols bgp out-delay *seconds*],
[edit protocols dvmrp],
[edit protocols isis],
[edit protocols ldp],
[edit protocols msdp],
[edit protocols msdp group *group-name*],
[edit protocols msdp group *group-name* peer *peer-address*],
[edit protocols msdp peer *address*],
[edit protocols ospf],
[edit protocols ospf3],
[edit protocols pim rp bootstrap family (inet | inet6)],
[edit rip group *group-name*],
[edit ripng group *group-name*],
[edit routing-instances *routing-instance-name* protocols bgp],
[edit routing-instances *routing-instance-name* protocols bgp group *group-name*],
[edit routing-instances *routing-instance-name* protocols bgp group *group-name* neighbor *address*],
[edit routing-instances *routing-instance-name* protocols bgp group *group-name* neighbor *address* out-delay *seconds*],
[edit routing-instances *routing-instance-name* protocols bgp out-delay *seconds*],
[edit routing-instances *routing-instance-name* protocols dvmrp],
[edit routing-instances *routing-instance-name* protocols isis],
[edit routing-instances *routing-instance-name* protocols ldp],
[edit routing-instances *routing-instance-name* protocols msdp],
[edit routing-instances *routing-instance-name* protocols msdp group *group-name*],
[edit routing-instances *routing-instance-name* protocols msdp group *group-name* peer *address*],
[edit routing-instances *routing-instance-name* protocols msdp peer *address*],
[edit routing-instances *routing-instance-name* protocols ospf],
[edit routing-instances *routing-instance-name* protocols ospf3],

[edit routing-instances *routing-instance-name* protocols pim rp bootstrap family (inet | inet6)],
[edit routing-instances *routing-instance-name* protocols rip group *group-name*],
[edit routing-instances *routing-instance-name* protocols ripng group *group-name*]

Release Information Statement introduced before Junos OS Release 7.4.

Description Apply one or more policies to routes being exported from the routing table into a routing protocol.

Options ***policy-names***—Names of one or more policies defined with a **policy-statement** statement.

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

Related Documentation

- Applying Routing Policies and Policy Chains to Routing Protocols on page 64

import

Syntax import [*policy-names*];

Hierarchy Level
[edit logical-systems *logical-system-name* protocols bgp],
[edit logical-systems *logical-system-name* protocols bgp group *group-name*],
[edit logical-systems *logical-system-name* protocols dvmrp],
[edit logical-systems *logical-system-name* protocols ldp],
[edit logical-systems *logical-system-name* protocols msdp],
[edit logical-systems *logical-system-name* protocols msdp peer *address*],
[edit logical-systems *logical-system-name* protocols msdp group *group-name*],
[edit logical-systems *logical-system-name* protocols msdp group *group-name* peer *address*],
[edit logical-systems *logical-system-name* protocols ospf],
[edit logical-systems *logical-system-name* protocols ospf3],
[edit logical-systems *logical-system-name* protocols pim],
[edit logical-systems *logical-system-name* protocols pim rp bootstrap family (inet | inet6)],
[edit logical-systems *logical-system-name* protocols rip],
[edit logical-systems *logical-system-name* protocols rip group *group-name*],
[edit logical-systems *logical-system-name* protocols rip group *group-name* neighbor *address*],
[edit logical-systems *logical-system-name* protocols ripng],
[edit logical-systems *logical-system-name* protocols ripng group *group-name*],
[edit logical-systems *logical-system-name* protocols ripng group *group-name* neighbor *address*],
[edit protocols bgp],
[edit protocols bgp group *group-name*],
[edit protocols bgp group *group-name* neighbor *address*],
[edit protocols dvmrp],
[edit protocols ldp],
[edit protocols msdp],
[edit protocols msdp peer *address*],
[edit protocols msdp group *group-name*],
[edit protocols msdp group *group-name* peer *address*],
[edit protocols ospf],
[edit protocols ospf3],
[edit protocols pim],
[edit protocols pim rp bootstrap family (inet | inet6)],
[edit protocols rip],
[edit protocols rip group *group-name*],
[edit protocols rip group *group-name* neighbor *address*],
[edit protocols ripng],
[edit protocols ripng group *group-name*],
[edit protocols ripng group *group-name* neighbor *address*],
[edit routing-instances *routing-instance-name* protocols bgp],
[edit routing-instances *routing-instance-name* protocols bgp group *group-name* neighbor *address*],
[edit routing-instances *routing-instance-name* protocols dvmrp],
[edit routing-instances *routing-instance-name* protocols ldp],
[edit routing-instances *routing-instance-name* protocols msdp],
[edit routing-instances *routing-instance-name* protocols msdp peer *address*],
[edit routing-instances *routing-instance-name* protocols msdp group *group-name*],
[edit routing-instances *routing-instance-name* protocols msdp group *group-name* peer *address*],
[edit routing-instances *routing-instance-name* protocols ospf],
[edit routing-instances *routing-instance-name* protocols ospf3],

[edit routing-instances *routing-instance-name* protocols pim],
[edit routing-instances *routing-instance-name* protocols pim rp bootstrap family (inet | inet6)],
[edit routing-instances *routing-instance-name* protocols rip],
[edit routing-instances *routing-instance-name* protocols rip group *group-name*],
[edit routing-instances *routing-instance-name* protocols rip group *group-name* neighbor *address*],
[edit routing-instances *routing-instance-name* protocols ripng],
[edit routing-instances *routing-instance-name* protocols ripng group *group-name*],
[edit routing-instances *routing-instance-name* protocols ripng group *group-name* neighbor *address*]

Release Information Statement introduced before Junos OS Release 7.4.

Description Apply one or more policies to routes being imported into the routing table from a routing protocol.

Options ***policy-names***—Names of one or more policies defined with a **policy-statement** statement.

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

Related Documentation
- Applying Routing Policies and Policy Chains to Routing Protocols on page 64

policy-options

Syntax policy-options { ... }

Hierarchy Level [edit],
[edit dynamic],
[edit dynamic-profiles *profile-name*]

Release Information Statement introduced before Junos OS Release 7.4.
Support at the **[edit dynamic-profiles]** hierarchy level introduced in Junos OS Release 11.4.

Description Configure routing policy.

The statements are explained separately.

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

Related Documentation
- Defining Routing Policies on page 46

policy-statement

Syntax

```
policy-statement policy-name {
  term term-name {
    from {
      family family-name;
      match-conditions;
      policy subroutine-policy-name;
      prefix-list prefix-list-name;
      prefix-list-filter prefix-list-name match-type <actions>;
      route-filter destination-prefix match-type <actions>;
      source-address-filter source-prefix match-type <actions>;
    }
    to {
      match-conditions;
      policy subroutine-policy-name;
    }
    then actions;
  }
}
```

Hierarchy Level

[edit dynamic policy-options],
[edit logical-systems *logical-system-name* policy-options],
[edit policy-options]

Release Information

Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.
Support for configuration in the dynamic database introduced in Junos OS Release 9.5.
Support for configuration in the dynamic database introduced in Junos OS Release 9.5 for EX Series switches.
inet-mdt option introduced in Junos OS Release 10.0R2.
Statement introduced in Junos OS Release 11.3 for the QFX Series.

Description

Define a routing policy, including subroutine policies.

To list the routing policies under the **[edit policy-options]** hierarchy level by **policy-statement *policy-name*** in alphabetical order, enter the **show policy-options** configuration command.

Options

actions—(Optional) One or more actions to take if the conditions match. The actions are described in "Configuring Flow Control Actions" on page 55.

family *family-name*—(Optional) Specify an address family protocol. Specify **inet** for IPv4. Specify **inet6** for 128-bit IPv6, and to enable interpretation of IPv6 router filter addresses. For IS-IS traffic, specify **iso**. For IPv4 multicast VPN traffic, specify **inet-mvpn**. For IPv6 multicast VPN traffic, specify **inet6-mvpn**. For multicast-distribution-tree (MDT) IPv4 traffic, specify **inet-mdt**.

NOTE: When **family** is not specified, the routing device uses the default IPv4 setting.

from—(Optional) Match a route based on its source address.

match-conditions—(Optional in **from** statement; required in **to** statement) One or more conditions to use to make a match. The qualifiers are described in "Configuring Match Conditions in Routing Policy Terms" on page 47.

policy ***subroutine-policy-name***—Use another policy as a match condition within this policy. The name identifying the subroutine policy can contain letters, numbers, and hyphens (-) and can be up to 255 characters long. To include spaces in the name, enclose it in quotation marks (" "). For information about how to configure subroutines, see "Configuring Subroutines in Routing Policy Match Conditions" on page 146.

policy-name—Name that identifies the policy. The name can contain letters, numbers, and hyphens (-) and can be up to 255 characters long. To include spaces in the name, enclose it in quotation marks (" ").

prefix-list ***prefix-list-name*** —Name of a list of IPv4 or IPv6 prefixes.

prefix-list-filter ***prefix-list-name***—Name of a prefix list to evaluate using qualifiers; ***match-type*** is the type of match (see "Configuring Prefix List Filters" on page 131), and ***actions*** is the action to take if the prefixes match.

route-filter ***destination-prefix match-type <actions>***—(Optional) List of routes on which to perform an immediate match; ***destination-prefix*** is the IPv4 or IPv6 route prefix to match, ***match-type*** is the type of match (see "Configuring Route Lists" on page 132), and ***actions*** is the action to take if the ***destination-prefix*** matches.

source-address-filter ***source-prefix match-type <actions>***—(Optional) Unicast source addresses in multiprotocol BGP (MBGP) and Multicast Source Discovery Protocol (MSDP) environments on which to perform an immediate match. ***source-prefix*** is the IPv4 or IPv6 route prefix to match, ***match-type*** is the type of match (see "Configuring Route Lists" on page 132), and ***actions*** is the action to take if the ***source-prefix*** matches.

term ***term-name***—Name that identifies the term.

to—(Optional) Match a route based on its destination address or the protocols into which the route is being advertised.

then—(Optional) Actions to take on matching routes. The actions are described in "Configuring Flow Control Actions" on page 55 and "Configuring Actions That Manipulate Route Characteristics" on page 56.

Required Privilege Level

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

Related Documentation

- Defining Routing Policies on page 46
- Configuring Routing Policies and Policy Objects in the Dynamic Database on page 71
- dynamic-db on page 242

prefix-list

Syntax

```
prefix-list name {
  ip-addresses;
  apply-path path;
}
```

Hierarchy Level

[edit dynamic policy-options],
[edit logical-systems *logical-system-name* policy-options],
[edit policy-options]

Release Information

Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.
Support for configuration in the dynamic database introduced in Junos OS Release 9.5.
Support for configuration in the dynamic database introduced in Junos OS Release 9.5 for EX Series switches.
Support for the **vpls** protocol family introduced in Junos OS Release 10.2.

Description

Define a list of IPv4 or IPv6 address prefixes for use in a routing policy statement or firewall filter statement.

You can configure up to 85,325 prefixes in each prefix list. To configure more than 85,325 prefixes, configure multiple prefix lists and apply them to multiple firewall filter terms.

Options

name—Name that identifies the list of IPv4or IPv6 address prefixes.

ip-addresses—List of IPv4 or IPv6 address prefixes, one IP address per line in the configuration.

The remaining statement is explained separately.

Required Privilege Level

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

Related Documentation

- Configuring Prefix Lists for Use in Routing Policy Match Conditions on page 127
- Configuring Routing Policies and Policy Objects in the Dynamic Database on page 71
- dynamic-db on page 242
- "Firewall Filter Match Conditions Based on Address Fields" in the *Junos OS Firewall Filter and Policer Configuration Guide*
- "Example: Configuring a Filter to Limit TCP Access to a Port Based On a Prefix List" in the *Junos OS Firewall Filter and Policer Configuration Guide*

prefix-list-filter

Syntax prefix-list-filter *prefix-list-name match-type* <*actions*>;

Hierarchy Level [edit logical-systems *logical-system-name* policy-options],
[edit policy-options]

Release Information Statement introduced before Junos OS Release 7.4.

Description Evaluate a list of prefixes within a prefix list using specified qualifiers.

Options ***prefix-list-name***—Name of the prefix list to evaluate.

match-type—Prefix length qualifiers.

actions—(Optional) Actions to take on match.

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

Related Documentation

- Configuring Prefix Lists for Use in Routing Policy Match Conditions on page 127

CHAPTER 15

Summary of Traffic Sampling, Forwarding, and Monitoring Configuration Statements

accounting

Syntax

```
accounting group-name {
  output {
    aggregate-export-interval seconds;
    cflowd [ hostnames ] {
      aggregation {
        autonomous-system;
        destination-prefix;
        protocol-port;
        source-destination-prefix {
          caida-compliant;
        }
        source-prefix;
      }
      autonomous-system-type (origin | peer);
      port port-number;
      version format;
    }
    flow-active-timeout seconds;
    flow-inactive-timeout seconds;
    interface interface-name {
      engine-id number;
      engine-type number;
      source-address address;
    }
  }
}
```

Hierarchy Level [edit forwarding-options]

Release Information Statement introduced before Junos OS Release 7.4.

Description Specify discard accounting instance name and options.

The statements are explained separately.

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

Related Documentation

- Configuring Discard Accounting on page 205

aggregation

Syntax

```
aggregation {
  autonomous-system;
  destination-prefix;
  protocol-port;
  source-destination-prefix {
    caida-compliant;
  }
  source-prefix;
}
```

Hierarchy Level

[edit forwarding-options accounting output *hostname*],
[edit forwarding-options sampling family (inet | inet6 | mpls) output flow-server *hostname*]

Release Information

Statement introduced before Junos OS Release 7.4.

Description

For cflowd version 8 only, specify the type of data to be aggregated; cflowd records and sends only those flows that match the specified criteria.

Options

autonomous-system—Aggregate by autonomous system (AS) number.

caida-compliant—Record source and destination mask length values in compliance with the Version 2.1b1 release of the cflowd application from the Cooperative Association for Internet Data Analysis (CAIDA). If this statement is not configured, the Junos OS records source and destination mask length values in compliance with the *cflowd Configuration Guide*, dated August 30, 1999.

destination-prefix—Aggregate by destination prefix.

protocol-port—Aggregate by protocol and port number.

source-destination-prefix—Aggregate by source and destination prefix.

source-prefix—Aggregate by source prefix.

Required Privilege Level

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

Related Documentation

- Configuring Flow Aggregation (cflowd) on page 189

autonomous-system-type

Syntax autonomous-system-type (origin | peer);

Hierarchy Level [edit forwarding-options accountingoutput cflowd *hostname*],
[edit forwarding-options sampling family (inet | inet6 | mpls)output flow-server *hostname*]

Release Information Statement introduced before Junos OS Release 7.4.

Description Specify the type of AS numbers that cflowd exports.

Default origin

Options **origin**—Export origin AS numbers of the packet source address in the Source Autonomous System cflowd field.

peer—Export peer AS numbers through which the packet passed in the Source Autonomous System cflowd field.

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

Related Documentation

- Configuring Flow Aggregation (cflowd) on page 189

bootp

Syntax

```
bootp {
  client-response-ttl number;
  description text-description;
  interface (interface-name | interface-group) {
    client-response-ttl number;
    description text-description;
    maximum-hop-count number;
    minimum-wait-time seconds;
    no-listen;
    server address {
      logical-system logical-system-name <routing-instance [ <default>
        routing-instance-names ]>;
      routing-instance [ <default> routing-instance-names ];
    }
  }
  maximum-hop-count number;
  minimum-wait-time seconds;
  relay-agent-option;
  server address {
    <logical-system logical-system-name> <routing-instance
      [ routing-instance-names ]>;
  }
}
```

Hierarchy Level [edit forwarding-options helpers]

Release Information Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.
Statement introduced in Junos OS Release 11.3 for QFX Series switches.

Description Configures a router, switch, or interface to act as a Dynamic Host Configuration Protocol (DHCP) or bootstrap protocol (BOOTP) relay agent.

DHCP relaying is disabled.

Options The remaining statements are explained separately.

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

Related Documentation

- Configuring Routers, Switches, and Interfaces as DHCP and BOOTP Relay Agents on page 210
- Setting Up DHCP Option 82 with the Switch as a Relay Agent Between Clients and DHCP Server (CLI Procedure)

cflowd (Discard Accounting)

Syntax

```
cflowd hostname {
  aggregation {
    autonomous-system;
    destination-prefix;
    protocol-port;
    source-destination-prefix {
      caida-compliant;
    }
    source-prefix;
  }
  autonomous-system-type (origin | peer);
  port port-number;
  version format;
}
```

Hierarchy Level [edit forwarding-options accounting *group-name* output]

Release Information Statement introduced before Junos OS Release 7.4.

Description Collect an aggregate of sampled flows and send the aggregate to a specified host system that runs the collection utility cfdcollect.

You can configure up to one version 5 and one version 8 flow format at the **[edit forwarding-options accounting *group-name* output]** hierarchy level.

Options ***hostname***—The IP address or identifier of the host system (the workstation running the cflowd utility).

The remaining statements are explained separately.

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

Related Documentation

- Configuring Flow Aggregation (cflowd) on page 189

cflowd (Flow Monitoring)

Syntax
```
cflowd hostname {
  port port-number;
}
```

Hierarchy Level [edit forwarding-options monitoring *group-name* family inet output]

Release Information Statement introduced before Junos OS Release 7.4.

Description Collect an aggregate of sampled flows and send the aggregate to a specified host system that runs the collection utility cfdcollect.

You can configure up to eight version 5 flow formats at the **[edit forwarding-options monitoring *group-name* output]** hierarchy level. Version 8 flow formats are not supported for flow-monitoring applications.

Options ***hostname***—The IP address or identifier of the host system (the workstation running the cflowd utility).

The remaining statements are explained separately.

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

Related Documentation
- Configuring Flow Monitoring on page 206

client-response-ttl

Syntax client-response-ttl *number*;

Hierarchy Level [edit forwarding-options helpers bootp],
[edit forwarding-options helpers bootp interface (*interface-name* | *interface-group*)]

Release Information Statement introduced in Junos OS Release 8.1.
Statement introduced in Junos OS Release 11.3 for QFX Series switches.

Description Set the IP time-to-live (TTL) value in DHCP response packets sent to a DHCP client.

Options ***number***—Decrement amount.

Default: None

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

Related Documentation
- Configuring Routers, Switches, and Interfaces as DHCP and BOOTP Relay Agents on page 210

description

Syntax description *text-description*;

Hierarchy Level [edit forwarding-options helpers bootp],
[edit forwarding-options helpers bootpinterface (*interface-name* | *interface-group*)],
[edit forwarding-options helpers domain],
[edit forwarding-options helpers domain interface *interface-name*],
[edit forwarding-options helpers tftp],
[edit forwarding-options helpers tftpinterface *interface-name*]

Release Information Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.
Statement introduced in Junos OS Release 11.3 for QFX Series switches.

Description Describe a BOOTP, DHCP, Domain Name System (DNS), or Trivial File Transfer Protocol (TFTP) service, or an interface that is configured for the service.

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

Related Documentation

- Configuring DNS and TFTP Packet Forwarding on page 212
- Configuring Routers, Switches, and Interfaces as DHCP and BOOTP Relay Agents on page 210

dhcp-relay (DHCP Spoofing Prevention)

Syntax

```
dhcp-relay {
  group group-name {
    interface interface-name;
  }
}
```

Hierarchy Level

[edit routing-instances *routing-instance-name* bridge-domains *bridge-domain-name* forwarding-options],
[edit routing-instances *routing-instance-name* forwarding-options]

Release Information

Statement introduced in Junos OS Release 9.4 (MX Series routers only).

Description

Configure Dynamic Host Configuration Protocol (DHCP) snooping on the router. When acting as a snooping agent, the MX Series router typically is located between the client and the DHCP relay agent. It creates filters by "snooping" DHCP messages and binding DHCP-issued IP addresses to the MAC address of the client. These filters help prevent DHCP spoofing.

Configure DHCP snooping by including the appropriate interfaces in the DHCP relay configuration.

The statements are explained separately.

Required Privilege Level

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

Related Documentation

- Preventing DHCP Spoofing on MX Series 3D Universal Edge Routers on page 215

disable

Syntax

```
disable;
```

Hierarchy Level

[edit forwarding-options packet-capture],
[edit forwarding-options sampling],
[edit forwarding-options sampling family *family-name* output file]

Release Information

Statement introduced before Junos OS Release 7.4.
Supported added at the **[edit forwarding-options packet-capture]** hierarchy level on J Series Services Routers in Junos OS Release 7.5.

Description

Disable traffic sampling or (on J Series Services Routers) packet capture.

Required Privilege Level

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

Related Documentation

- Configuring Packet Capture on page 221
- Disabling Traffic Sampling on page 187

domain

Syntax

```
domain {
  description text-description;
  interface interface-name {
    broadcast;
    description text-description;
    no-listen;
    server address <logical-system logical-system-name> <routing-instance
      routing-instance-name>;
  }
  server address <logical-system logical-system-name> <routing-instance
    routing-instance-name>;
}
```

Hierarchy Level [edit forwarding-options helpers]

Release Information Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Enable DNS request packet forwarding.

The remaining statements are explained separately.

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

Related Documentation
- Configuring DNS and TFTP Packet Forwarding on page 212

export-format

Syntax export-format cflowd-version-5;

Hierarchy Level [edit forwarding-options monitoring *group-name* family inet output]

Release Information Statement introduced before Junos OS Release 7.4.

Description Flow monitoring export format.

Options **cflowd-version-5**—cflowd version 5.

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

Related Documentation
- Configuring Flow Monitoring on page 206

enhanced-hash-key

Syntax

```
enhanced-hash-key {
  family [
    inet
      incoming-interface-index;
        no-destination-port;
        no-source-port;
        type-of-service;
      }
    inet6 {
      incoming-interface-index;
      no-destination-port;
      no-source-port;
      traffic-class;
    }
    mpls {
      incoming-interface-index;
      label-1-exp;
      no-payload;
    }
    multiservice {
      incoming-interface-index;
      no-payload;
      outer-priority;
    }
  }
  services-loadbalancing {
    family inet layer-3-services {
      incoming-interface-index;
      source-address;
    }
  }
}
```

Hierarchy Level [edit forwarding-options],
[edit logical-systems logical-system-name routing-instances instance-name forwarding-options],
[edit routing-instances instance-name forwarding-options]

Release Information Statement introduced in Junos OS Release 10.1.
services-loadbalancing option introduced in Junos OS Release 11.2.

Description For MX Series routers with MPCs, select data used in the hash key for enhanced IP forwarding engines.

Default Not enabled.

Options **services-loadbalancing**—Distributes traffic across PICs based on source IP address when a route pointing to more than one services PICs is installed.

Data selections for **services-loadbalancing**:

- **inet**—IPv4 addressing protocol.
- **layer-3-services**—Include layer 3 IP data in the hash key.
- **incoming-interface-index**—Include incoming interface index in the hash key.
- **source-address**—Include source-address in the hash key.

Data selections for family **inet**:

- **incoming-interface-index**—Include incoming interface index in the hash key.
- **no-destination-port**—Omit IP destination port in the hash key.
- **no-source-port**—Omit IP source port in the hash key..
- **type-of-service**—Include type-of-service (TOS) byte in the hash key.

Data selections for family **inet6**:

- **incoming-interface-index**—Include the incoming interface index in the hash key.
- **no-destination-port**—Omit the IP destination port in the hash key.
- **no-source-port**—Omit the IP source port in the hash key.
- **traffic-class**—Include the traffic class byte in the hash key.

Data selections for family **mpls**:

- **incoming-interface-index**—Include the incoming interface index in the hash key.
- **label-1-exp**—The EXP bit of the first label is used in the hash calculation.
- **no-payload**—Omit the MPLS payload data from the hash key.

Data selections for family **multiservice**:

- **incoming-interface-index**—Include the incoming interface index in the hash key.
- **no-payload**—Omit the payload data from the hash key.
- **outer-priority**—Include the outer 802.1 priority bits in the hash key.

Required Privilege Level

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

Related Documentation

- [edit forwarding-options] Hierarchy Level

family (Filtering)

Syntax

```
family family-name {
  filter {
    input input-filter-name;
    output output-filter-name;
  }
  flood {
    input filter-name;
  }
  route-accounting;
}
```

Hierarchy Level [edit forwarding-options]

Release Information Statement introduced before Junos OS Release 7.4.
route-accounting option introduced in Junos OS Release 8.3; supported only with IPv6.

Description Specify address family for filters.

Options *family-name*—Address family. Specify **inet** for IP version 4 (IPv4), **inet6** for IP version 6 (IPv6), **mpls** for MPLS, or **vpls** for virtual private LAN service (VPLS).

NOTE: In Junos OS Release 8.4 and later, the **output** statement is not valid at the **[edit forwarding-options family vpls filter]** hierarchy level.

The remaining statements are explained separately.

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

Related Documentation

- Applying Filters to Forwarding Tables on page 203

family (Monitoring)

Syntax

```
family inet {
  output {
    cflowd\ hostname {
      port port-number;
    }
    export-format cflowd-version-5;
    flow-active-timeout seconds;
    flow-export-destination {
      (cflowd-collector | collector-pic);
    }
    flow-inactive-timeout seconds;
    interface interface-name {
      engine-id number;
      engine-type number;
      input-interface-index number;
      output-interface-index number;
      source-address address;
    }
  }
}
```

Hierarchy Level [edit forwarding-options monitoring *group-name*]

Release Information Statement introduced before Junos OS Release 7.4.

Description Configure flow monitoring for an address family. Only the IPv4 protocol is supported.

The remaining statements are explained separately.

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

Related Documentation

- Configuring Flow Monitoring on page 206

family (Port Mirroring)

Syntax

```
family (ccc | inet | inet6 | vpls) {
  output {
    interface interface-name {
      next-hop address;
    }
    no-filter-check;
  }
}
```

Hierarchy Level

[edit forwarding-options port-mirroring],
[edit forwarding-options port-mirroring instance *instance-name*]

Release Information

Statement introduced before Junos OS Release 7.4.
vpls option introduced in Junos OS Release 9.3 for MX Series routers only; support extended to M7i, M10i, M120, and M320 routers in Junos OS Release 9.5.
ccc option introduced in Junos OS Release 9.6 for M120 and M320 routers only.

Description

Configure the address type family to sample for port mirroring.

Options

ccc—Sample Layer 2 VPN traffic.

inet—Sample IPv4 traffic.

inet6—Sample IPv6 traffic.

vpls—Sample VPLS traffic

The remaining statements are explained separately.

Required Privilege Level

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

Related Documentation

- Configuring Port Mirroring on page 217

family (Sampling)

Syntax

```
family (inet | inet6 | mpls) {
  disable;
  output {
    aggregate-export-interval seconds;
    extension-service service-name;
    file {
      disable;
      filename filename;
      files number;
      size bytes;
      (stamp | no-stamp);
      (world-readable | no-world-readable);
    }
    flow-active-timeout seconds;
    flow-inactive-timeout seconds;
    flow-server hostname {
      aggregation {
        autonomous-system;
        destination-prefix;
        protocol-port;
        source-destination-prefix {
          caida-compliant;
        }
        source-prefix;
      }
      autonomous-system-type (origin | peer);
      (local-dump | no-local-dump);
      port port-number;
      source-address address;
      version format;
      version9 {
        template template-name;
      }
    }
    interface interface-name {
      engine-id number;
      engine-type number;
      source-address address;
    }
  }
}
```

Hierarchy Level [edit forwarding-options sampling]

Release Information Statement introduced before Junos OS Release 7.4.
mpls option introduced in Junos OS Release 8.3.

Description Configure the protocol family to be sampled.

Options **inet**—IP version 4 (IPv4)

inet6—IP version 6 (IPv6)

mpls—MPLS

The remaining statements are explained separately.

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

Related Documentation
- Configuring Traffic Sampling on page 185

family inet

Syntax

```
family inet {
  layer-3;
  layer-4;
}
```

Hierarchy Level
[edit forwarding-options hash-key]

Release Information
Statement introduced before Junos OS Release 7.4.

Description
Configure layer information for the load-balancing specification. Only the IPv4 protocol is supported.

Options
layer-3—Incorporate Layer 3 (IP) data into the hash key. When you select **layer-3**, the router uses following Layer 3 information to load-balance traffic:

- Source IP address
- Destination IP address
- Protocol
- Incoming interface index

layer-4—Incorporate Layer 4 Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) data into the hash key. When you select **layer-4**, the router uses following Layer 4 information to load-balance traffic:

- Destination port number
- Source port number
- IP type of service

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

Related Documentation
- Configuring Per-Packet Load Balancing on page 160

family mpls

Syntax

```
family mpls {
  label-1;
  label-2;
  label-3;
  no-labels;
  no-label-1-exp;
  payload {
    ether-pseudowire;
    ip {
      layer-3-only;
      port-data {
        source-msb;
        source-lsb;
        destination-msb;
        destination-lsb;
      }
    }
  }
}
```

Hierarchy Level [edit forwarding-options hash-key]

Release Information Statement introduced before Junos OS Release 7.4.
no-label-1-exp option introduced in Junos OS Release 8.0.
label-3 and **no-labels** options introduced in Junos OS Release 8.1.
ether-pseudowire statement introduced in Junos OS Release 9.1 (M320 and T Series routers only); support extended to M120 and MX Series routers in Junos OS Release 9.4.

Description For aggregated Ethernet and SONET/SDH interfaces only, configure load balancing based on MPLS labels and payload. Only the IPv4 protocol is supported.

Options

label-1—Use this to include the first MPLS label in the hash key. Used for one-label packet.

label-2—Use this to include the second MPLS label in the hash key. Configure this when you want to include the second MPLS label in the hash key. If both **label-1** and **label-2** are specified, the entire first label and the first 16 bits of the second label are hashed.

label-3—Use this to include the third MPLS label in the hash key. You must configure the **label-1**, **label-2**, and **label-3** statements in order to include the third label.

no-labels—Use this to omit MPLS labels from the hash key.

no-label-1-exp—Use this to omit the EXP bit of the first label from the hash calculation. Use this to avoid complications from reordering.

payload—Use this to incorporate bits from the IP payload in the hash key.

ether-pseudowire (M120, M320, MX Series, and T Series routers)—Load-balance IPv4 traffic over Layer 2 Ethernet pseudowires.

ip—Use this to include the IP address of the IPv4 or IPv6 payload in the hash key.

layer-3-only—Use this to include only Layer 3 IP information from the IP payload data.

port-data—Use this to include the source and destination port field information. By default, the most significant byte and least significant byte of the source and destination port fields are hashed. To select specific bytes to be hashed, include one or more of the **source-msb**, **source-lsb**, **destination-msb**, and **destination-lsb** options at the **[edit forwarding-options hash-key family mpls payload ip port-data]** hierarchy level. To prevent all four bytes from being hashed, include the **layer-3-only** statement at the **[edit forwarding-options hash-key family mpls payload ip]** hierarchy level.

destination-lsb—Use this to include the least significant byte of the destination port.

destination-msb—Use this to include the most significant byte of the destination port.

source-lsb—Use this to include the least significant byte of the source port.

source-msb—Use this to include the most significant byte of the source port.

Required Privilege Level

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

Related Documentation

- Configuring Load Balancing Based on MPLS Labels on page 163
- Configuring Load Balancing for Ethernet Pseudowires on page 166

family multiservice

Syntax

```
family multiservice {
  destination-mac;
  label-1;
  label-2;
  payload {
    ip {
      layer-3 {
        (destination-ip-only | source-ip-only);
      }
      layer-3-only;
      layer-4;
    }
  }
  source-mac;
  symmetric-hash {
    complement;
  }
}
```

Hierarchy Level [edit forwarding-options hash-key]

Release Information Statement introduced in Junos OS Release 8.0.
ip, **label-1**, **label-2**, **layer-3-only**, and **payload** statements introduced in Junos OS Release 9.4.
layer-3, **layer-**. **source-address-only**, and **destination-address-only** statements introduced in Junos OS Release 9.5.
symmetric-hash statement and the **complement** option introduced in Junos OS Release 9.6.

Description (M Series, MX Series, and T Series routers only) Configure load balancing based on Layer 2 media access control information. On MX Series routers, configure VPLS load balancing. On M120 and M320 routers only, configure VPLS load balancing based on MPLS labels and IP information. For IPv4 traffic, only the IP source and destination addresses are included in the hash key. For MPLS and IPv4 traffic, one or two MPLS labels and IPv4 source and destination addresses are included. For MPLS Ethernet pseudowires, only one or two MPLS labels are included in the hash key.

Options You can configure one or more options to load-balance using the packet information that you specify.

destination-mac—Configure this when you want to include the destination-address MAC information in the hash key for Layer 2 load balancing.

source-mac—Configure this when you want to include the source-address MAC information in the hash key.

label-1 (M120 and M320 routers only)—Configure this when you want to include the first MPLS label in the hash key. Used for including a one-label packet for per-flow load balancing of IPv4 VPLS traffic based on IP information and MPLS labels.

label-2 (M120 and M320 routers only)—Configure this when you want to include the second MPLS label in the hash key. If both **label-1** and **label-2** are specified, the entire first label and the first 16 bits of the second label are hashed.

payload (MX Series, M120, and M320 routers only)—Use this to include the packets' IP payload in the hash key.

ip (MX Series, M120, and M320 routers only)—Use this to include the IP address of the IPv4 or IPv6 payload in the hash key.

layer-3-only (M120, and M320 routers only)—Use this to include only the Layer 3 information from the packets' IP payload in the hash key.

layer-3 (MX Series routers only)—Use this to include Layer 3 information from the packets' IP payload in the hash key.

source-address-only (MX Series routers only)—Use this to include only the source IP address in the payload in the hash key.

destination-address-only (MX Series routers only)—Use this to include only the destination IP address in the payload in the hash key.

NOTE: You can include either the **source-address-only** or the **destination-address-only** statement, not both. They are mutually exclusive.

layer-4 (MX Series routers only)—Include Layer 4 information from the packets' IP payload in the hash key.

NOTE: On MX Series routers only, you can configure either Layer 3 or Layer 4 load balancing, or both at the same time.

NOTE: On I chip platforms, an unknown Layer 4 header is excluded from load balance hashing to avoid undesired packet reordering.

symmetric-hash (MX Series 3D Universal Edge Routers only)—Configure the symmetric hash or symmetric hash complement for configuring symmetrical load balancing on an 802.3ad Link Aggregation Group.

complement (MX Series 3D Universal Edge Routers only)—Include the complement of the symmetric hash in the hash key.

Required Privilege Level

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

Related Documentation

- Configuring Load Balancing Based on MAC Addresses on page 167
- Configuring VPLS Load Balancing Based on IP and MPLS Information on page 168
- Configuring VPLS Load Balancing on MX Series 3D Universal Edge Routers on page 169
- Configuring VPLS Load Balancing

file (Extended DHCP Relay Agent and Helpers Trace Options)

Syntax file *filename* <files *number*> <match *regular-expression*> <size *bytes*> <world-readable | no-world-readable>;

Hierarchy Level [edit forwarding-options dhcp-relay traceoptions],
[edit forwarding-options helpers traceoptions]

Release Information Statement introduced before Junos OS Release 7.4.

Description Configure information about the DNS and TFTP packet-forwarding files that contain trace logging information.

Options *filename*—Name of the file containing the trace information.

Default: /var/log/sampled

The remaining statements are explained separately.

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

Related Documentation

- Tracing BOOTP, DNS, and TFTP Forwarding Operations on page 213

file (Packet Capture)

Syntax file filename *filename* <files *number*> <size *bytes*> <world-readable | no-world-readable>;

Hierarchy Level [edit forwarding-options packet-capture]

Release Information Statement introduced in Junos OS Release 7.5.

Description Enable packet capture to a file.

Options The statements are explained separately.

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

Related Documentation

- Configuring Packet Capture on page 221

file (Sampling)

Syntax file filename *filename* <disable> <files *number*> <stamp | no-stamp> <size *bytes*> <world-readable | no-world-readable>;

Hierarchy Level [edit forwarding-options sampling family *family-name* output]

Release Information Statement introduced before Junos OS Release 7.4.

Description Collect the traffic samples in a file.

The statements are explained separately.

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

Related Documentation
- Configuring the Output File for Traffic Sampling on page 187

file (Trace Options)

Syntax file *filename* <files *number*> <size *bytes*> <world-readable | no-world-readable>;

Hierarchy Level [edit forwarding-options port-mirroring traceoptions],
[edit forwarding-options sampling traceoptions]

Release Information Statement introduced before Junos OS Release 7.4.

Description Configure information about the files that contain trace logging information.

Options *filename*—The name of the file containing the trace information.

Default: /var/log/sampled

The remaining statements are explained separately.

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

Related Documentation
- Tracing Traffic-Sampling Operations on page 189

filename (Packet Capture)

Syntax filename *filename*;

Hierarchy Level [edit forwarding-options packet-capture file]

Release Information Statement introduced in Junos OS Release 7.5.

Description Configure the name of the output file.

Options ***filename***—Name of the file.

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

Related Documentation
- Configuring Packet Capture on page 221

filename (Sampling)

Syntax filename *filename*;

Hierarchy Level [edit forwarding-options sampling family *family-name* output file]

Release Information Statement introduced before Junos OS Release 7.4.

Description Configure the name of the output file.

Options ***filename***—Name of the file in which to place the traffic samples. All files are placed in the directory **/var/tmp**.

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

Related Documentation
- Configuring the Output File for Traffic Sampling on page 187

files (Packet Capture)

Syntax files *number*;

Hierarchy Level [edit forwarding-options packet-capture file]

Release Information Statement introduced in Junos OS Release 7.5.

Description Configure the maximum number of files for packet capturing.

Options *number*—Maximum number of files.
Range: 2 through 10,000 files
Default: 10

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

Related Documentation
- Configuring Packet Capture on page 221

files (Sampling and Traceoptions)

Syntax files *number*;

Hierarchy Level [edit forwarding-options helperstraceoptions file],
[edit forwarding-options port-mirroring traceoptions file],
[edit forwarding-options sampling family *family-name* output file],
[edit forwarding-options sampling traceoptions file]

Release Information Statement introduced before Junos OS Release 7.4.

Description Configure the total number of files to be saved with samples or trace data.

Options *number*—Maximum number of traffic sampling or trace log files. When a file named ***sampling-file*** reaches its maximum size, it is renamed ***sampling-file*.0**, then ***sampling-file*.1**, and so on, until the maximum number of traffic sampling files is reached. Then the oldest sampling file is overwritten.
Range: 1 through 100 files
Default: 5 files for sampling output; 10 files for trace log information

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

Related Documentation
- Configuring the Output File for Traffic Sampling on page 187
- Tracing Traffic-Sampling Operations on page 189

filter (IPv4, IPv6, and MPLS)

Syntax

```
filter {
  input input-filter-name;
  output output-filter-name;
}
```

Hierarchy Level [edit forwarding-options family (inet | inet6 | mpls)]

Release Information Statement introduced before Junos OS Release 7.4.

Description Apply a forwarding table filter to a forwarding table.

Options The statements are explained separately.

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

Related Documentation

- Applying Filters to Forwarding Tables on page 203

filter (VPLS)

Syntax filter input *filter-name*;

Hierarchy Level [edit forwarding-options family vpls]

Release Information Statement introduced before Junos OS Release 7.4.

Description Apply a forwarding table filter for VPLS.

Options The other statement is explained separately.

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

Related Documentation

- Applying Filters to Forwarding Tables on page 203

flood

Syntax
```
flood {
  input filter-name;
}
```

Hierarchy Level [edit forwarding-optionsfamily vpls]

Release Information Statement introduced before Junos OS Release 7.4.

Description Apply a forwarding table filter to a flood table.

Options input *filter-name*—Name of the forwarding table filter.

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

Related Documentation
- Applying Filters to Forwarding Tables on page 203

flow-active-timeout

Syntax flow-active-timeout *seconds*;

Hierarchy Level [edit forwarding-options accounting *group-name* output],
[edit forwarding-options monitoring *group-name* family inet output],
[edit forwarding-optionssampling family *family-name* output]

Release Information Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Configure the time that elapses before another active flow is exported.

Options *seconds*—Timeout, in seconds.

Range: 60 through 1800

Default: 1800

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

Related Documentation
- Configuring Discard Accounting on page 205
- Configuring Flow Monitoring on page 206
- Configuring the Output File for Traffic Sampling on page 187

flow-export-destination

Syntax

```
flow-export-destination {
  (cflowd-collector | collector-pic);
}
```

Hierarchy Level [edit forwarding-options monitoring *group-name* family inet output]

Release Information Statement introduced before Junos OS Release 7.4.

Description Configure flow collection.

Options

cflowd-collector—cflowd collector.

collector-pic—Collector PIC.

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

Related Documentation

- Configuring Flow Monitoring on page 206

flow-inactive-timeout

Syntax flow-inactive-timeout *seconds*;

Hierarchy Level
[edit forwarding-options accounting *group-name* output],
[edit forwarding-options monitoring *group-name* family inet output],
[edit forwarding-options sampling family *family-name* output]

Release Information
Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Configure the time that elapses before a flow is considered inactive.

Options

seconds—Timeout, in seconds.

Range: 15 through 1800

Default: 60

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

Related Documentation

- Configuring Discard Accounting on page 205
- Configuring Flow Monitoring on page 206
- Configuring the Output File for Traffic Sampling on page 187

flow-server

Syntax

```
flow-server hostname {
  aggregation {
    autonomous-system;
    destination-prefix;
    protocol-port;
    source-destination-prefix {
      caida-compliant;
    }
    source-prefix;
  }
  autonomous-system-type (origin | peer);
  (local-dump | no-local-dump);
  port port-number;
  source-address address;
  version format;
  version9 {
    template template-name;
  }
}
```

Hierarchy Level [edit forwarding-options sampling family *family-name* output]

Release Information Statement introduced before Junos OS Release 7.4.
version9 statement introduced in Junos OS Release 8.3.

Description Collect an aggregate of sampled flows and send the aggregate to a specified host system that runs the collection utility cfdcollect. Specify a host system to collect sampled flows using the version 9 format.

You can configure up to one version 5 and one version 8 flow format at the **[edit forwarding-options sampling output flow-server *hostname*]** hierarchy level. For the same configuration, you can specify only either version 9 flow record formats or formats using versions 5 and 8, not both types of formats.

Options ***hostname***—The IP address or identifier of the host system (the workstation either running the cflowd utility or collecting traffic flows using version 9).

You can configure only one host system for version 9.

The remaining statements are explained separately.

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

Related Documentation

- Configuring Active Flow Monitoring Using Version 9 on page 192
- Configuring Flow Aggregation (cflowd) on page 189

forwarding-options

Syntax forwarding-options { ... }

Hierarchy Level [edit]

Release Information Statement introduced before Junos OS Release 7.4.

Description Configure traffic forwarding.

The statements are explained separately.

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

group (DHCP Spoofing Prevention)

Syntax
```
group group-name {
    interface interface-name;
}
```

Hierarchy Level [edit routing-instances *routing-instance-name* bridge-domains *bridge-domain-name* forwarding-options dhcp-relay],
[edit routing-instances *routing-instance-name* forwarding-options dhcp-relay]

Release Information Statement introduced in Junos OS Release 9.4 (MX Series routers only).

Description Configure Dynamic Host Configuration Protocol (DHCP) snooping on the router. When acting as a snooping agent, the MX Series router typically is located between the client and the DHCP relay agent. It creates filters by “snooping” DHCP messages and binding DHCP-issued IP addresses with the MAC address of the client. These filters help prevent DHCP spoofing.

Configure DHCP snooping by including the appropriate interfaces under the **group** statement.

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

Related Documentation
- Preventing DHCP Spoofing on MX Series 3D Universal Edge Routers on page 215

hash-key

Syntax

```
hash-key {
  family inet {
    layer-3;
    layer-4;
    symetric-hash;
  }
  family mpls {
    label-1;
    label-2;
    label-3;
    no-labels;
    no-label-1-exp;
    payload {
      ether-pseudowire;
      ip {
        layer-3-only;
        port-data {
          destination-lsb;
          destination-msb;
          source-lsb;
          source-msb;
        }
      }
    }
  }
  family multiservice {
    destination-mac;
    label-1;
    label-2;
    payload {
      ip {
        layer-3-only;
        layer-3 {
          (source-address-only | destination-address-only);
        }
        layer-4;
      }
    }
    source-mac:
  }
}
```

Hierarchy Level [edit forwarding-options]

Release Information Statement introduced before Junos OS Release 7.4.
family multiservice and **no-label-1-exp** options introduced in Junos OS Release 8.0.
label-3 and **no-labels** options introduced in Junos OS Release 8.1.
ether-pseudowire statement introduced in Junos OS Release 9.1 (M320 and T Series routers only); support extended to M120 and MX Series routers in Junos OS Release 9.4.

ip, **label-1**, **label-2**, **layer-3-only**, and **payload** options for the **family multiservice** statement introduced in Junos OS Release 9.4. (M120 and M320 routers only); support for **ip** and **payload** statements only to MX Series routers.
layer-3, **source-address-only**, **destination-address-only**, and **layer-4** statements introduced for the **family multiservice** statement in Junos OS Release 9.5. (MX Series routers only)

Description Select which packet header data to use for per-flow load balancing.

The options are explained separately.

NOTE: To modify the default hashing mechanism on Modular Port Concentrators (MPCs), you need to configure the statements at the **[edit forwarding-options enhanced-hash-key]** hierarchy level. Statements at the **[edit forwarding-options hash-key]** hierarchy level do not support MPCs.

Options

family inet—Incorporate port data into the hash key for flow determination. By default, port data is ignored when determining flows.

layer-3—Include Layer 3 data into the hash key. You must include the **layer-3** statement. If you omit the **layer-3** statement, the management process removes the **hash-key** statement from the configuration and the router behaves as if you specified **layer-3**.

By default, or if you specify only the **layer-3** statement, the router uses the following Layer 3 information in the packet header for per-flow load balancing:

- Source IP address
- Destination IP address
- Protocol

layer-4—Include Layer 4 data into the hash key.

If you include both the **layer-3** and **layer-4** statements, the router uses the following Layer 3 and Layer 4 information to load-balance:

- Source IP address
- Destination IP address
- Protocol
- Source port number
- Destination port number
- Incoming interface index

family mpls—(aggregated Ethernet interfaces, aggregated SONET/SDH interfaces, and multiple equal-cost MPLS next hops only) Incorporate multiprotocol label switching (MPLS) label and payload information into the hash key for per-flow load balancing. Only the IPv4 protocol is supported.

label-1—Include the first MPLS label into the hash key. This is used for a one-label packet for per-flow load balancing IPv4 VPLS traffic based on IP information and MPLS labels.

label-2—Include the second MPLS label into the hash key. This is used for a two-label packet for per-flow load balancing IPv4 VPLS traffic based on IP information and MPLS labels. To use the second MPLS label in the hash key, include both the **label-1** and **label-2** statements at the **[edit forwarding-options hash-key family mpls]** hierarchy level. By default, the router provides hashing on the first and second labels. If both labels are specified, the entire first label and the first 16 bits of the second label are hashed.

label-3—Include the third MPLS label into the hash key. To use the third MPLS label, include the **label-1**, **label-2**, and **label-3** statements at the **[edit forwarding-options hash-key family mpls]** hierarchy level.

no-labels—Include no MPLS labels into the hash key.

no-label-1-exp—The EXP bit of the first label is not used in the hash calculation to avoid reordering complication.

payload—Incorporate payload data into the hash key for per-flow load balancing Layer 2 information based on MPLS labels.

ether-pseudowire—(M120, M320, MX Series, and T Series routers)—Load-balance IPv4 traffic over Layer 2 Ethernet pseudowires.

ip—Include the IP address of the IPv4 or IPv6 payload into the hash key for per-flow load balancing Layer 2 information based on MPLS labels.

layer-3-only—Include only Layer 3 IP information from the IP payload data into the hash key for per-flow load balancing Layer 2 information based on MPLS labels.

port-data—Include the source and destination port field information into the hash key. By default, the most significant byte and least significant byte of the source and destination port fields are hashed. To select specific bytes to be hashed, include one or more of the **source-msb**, **source-lsb**, **destination-msb**, and **destination-lsb** options at the **[edit forwarding-options hash-key family mpls payload ip port-data]** hierarchy level. To prevent all four bytes from being hashed, include the **layer-3-only** statement at the **[edit forwarding-options hash-key family mpls payload ip]** hierarchy level.

destination-lsb—Include the least-significant byte of the destination port.

destination-msb—Include the most-significant byte of the destination port.

source-lsb—Include the least-significant byte of the source port.

source-msb—Include the most-significant byte of the source port.

family multiservice—(M Series, MX Series, and T Series Routers only) Include Layer 2 media access control (MAC) information into the hash key for per-flow load balancing only Layer 2 information.

family multiservice—(M120 and M320 Routers only) Include IP information and MPLS labels into the hash key for per-flow load balancing VPLS traffic. For IPv4 traffic, only the IP source and destination addresses are included in the hash key. For MPLS and IPv4 traffic, one or two MPLS labels and IPv4 source and destination addresses are included. For MPLS Ethernet psuedowires, only one or two MPLS labels are included in the hash key.

destination-mac—Include the destination MAC address into the hash key for per-flow load balancing Layer 2 information based on MPLS labels.

source-address-only—Include only the Layer 3 IP source address into the hash key for per-flow load balancing Layer 2 information based on MPLS labels.

destination-address-only—Include only the Layer 3 IP destination address into the hash key for per-flow load balancing Layer 2 information based on MPLS labels.

source-mac—Include the source-address MAC information into the hash key.

Required Privilege Level

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

Related Documentation

- Configuring Per-Packet Load Balancing on page 160
- Configuring Load Balancing Based on MPLS Labels on page 163
- Configuring Load Balancing Based on MAC Addresses on page 167

helpers

Syntax

```
helpers {
    bootp {
        client-response-ttl number;
        description text-description;
        interface interface-group {
            client-response-ttl number;
            description text-description;
            maximum-hop-count number;
            minimum-wait-time seconds;
            no-listen;
            server address {
                logical-system logical-system-name <routing-instance [ <default>
                    routing-instance-names ]>;
                routing-instance [ <default> routing-instance-names ];
            }
        }
        maximum-hop-count number;
        minimum-wait-time seconds;
        relay-agent-option;
        server address {
            logical-system logical-system-name <routing-instance [ <default>
                routing-instance-names ]>;
            routing-instance [ <default> routing-instance-names ];
        }
    }
    domain {
        description text-description;
        interface interface-name {
            broadcast;
            description text-description;
            no-listen;
            server address <logical-system logical-system-name> <routing-instance
                routing-instance-name>;
        }
        server address <logical-system logical-system-name> <routing-instance
            routing-instance-name>;
    }
    port port-number {
        description text-description;
        interface interface-name {
            broadcast;
            description text-description;
            no-listen;
            server address <logical-system logical-system-name> <routing-instance
                routing-instance-name>;
        }
        server address <logical-system logical-system-name> <routing-instance
            routing-instance-name>;
    }
    tftp {
        description text-description;
        interface interface-name {
```

```
        broadcast;
        description text-description;
        no-listen;
        server address <logical-system logical-system-name> <routing-instance
          routing-instance-name>;
      }
      server address <logical-system logical-system-name> <routing-instance
        routing-instance-name>;
    }
    traceoptions {
      file filename <files number> <match regular-expression> <size bytes> <world-readable |
        no-world-readable>;
      flag flag;
      level level;
      no-remote-trace level;
    }
}
```

Hierarchy Level [edit forwarding-options]

Release Information Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Enable TFTP or DNS request packet forwarding, or configure the router, switch, or interface to act as a DHCP/BOOTP relay agent. Use only one server address per interface or global configuration.

The remaining statements are explained separately.

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

Related Documentation

- Configuring DNS and TFTP Packet Forwarding on page 212
- Configuring Routers, Switches, and Interfaces as DHCP and BOOTP Relay Agents on page 210

indexed-load-balance

Syntax indexed-load-balance;

Hierarchy Level [edit forwarding-options load-balance],
[edit logical-systems *logical-system-name* forwarding-options load-balance],
[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* forwarding-options load-balance],
[edit routing-instances *routing-instance-name* forwarding-options load-balance]

Release Information Statement introduced in Junos OS Release 9.0.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Improve load-balance distribution for unicast and aggregated next hops. Include this statement if you notice issues with load-balance distribution for IPv4 traffic. The **indexed-load-balance** statement causes the creation of a nexthop structure that is not a function of the hash only, but is also a function of the low-order bits of the IP address.

CAUTION: Including the **indexed-load-balance** statement causes an increase in memory usage on the device.

Default Disabled

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

Related Documentation

- Configuring Per-Prefix Load Balancing on page 209

input (Forwarding Table)

Syntax input *filter-name*;

Hierarchy Level [edit forwarding-options family (inet | inet6 | mpls | vpls) filter],
[edit routing-instances *routing-instance-name* forwarding-options family (inet | inet6 | mpls | vpls) filter]

Release Information Statement introduced before Junos OS Release 7.4.

Description Apply a forwarding table filter to ingress traffic of the forwarding table.

Options ***filter-name***—Name of the applied filter.

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

Related Documentation
- Applying Filters to Forwarding Tables on page 203

input (Port Mirroring)

Syntax
```
input {
  maximum-packet-length bytes;
  rate number;
  run-length number;
}
```

Hierarchy Level [edit forwarding-options port-mirroring],
[edit forwarding-options port-mirroring instance *instance-name*]

Release Information Statement introduced before Junos OS Release 7.4.
maximum-packet-length option introduced in Junos OS Release 9.6 for M120 and M320 routers only.

Description Configure input packet properties for port mirroring.

The statements are explained separately.

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

Related Documentation
- Configuring Port Mirroring on page 217

input (Sampling)

Syntax

```
input {
  max-packets-per-second number;
  maximum-packet-length bytes;
  rate number;
  run-length number;
}
```

Hierarchy Level [edit forwarding-options sampling]

Release Information Statement introduced before Junos OS Release 7.4.
Support for sampling of MPLS traffic introduced in Junos OS Release 8.3.

Description Configure traffic sampling on a logical interface.

The statements are explained separately.

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

Related Documentation

- Configuring Traffic Sampling on page 185

instance

Syntax

```
instance {
  instance-name {
    input {
      maximum-packet-length bytes;
      rate number;
      run-length number;
    }
    family (ccc| inet | inet6 | vpls) {
      output {
        interface interface-name {
          next-hop address;
        }
        no-filter-check;
      }
    }
  }
}
```

Hierarchy Level [edit forwarding-options port-mirroring]

Release Information Statement introduced in Junos OS Release 9.3 (MX Series routers only).
Support extended to M120 and M320 routers in Junos OS Release 9.5.
maximum-packet-length and **ccc** options introduced in Junos OS Release 9.6 for M120 and M320 routers only.

Description Configure a port-mirroring instance.

Options *port-mirroring-instance-name*—Name of the port-mirroring instance.

The remaining statements are explained separately.

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

Related Documentation

- Configuring Port Mirroring on page 217

interface (Accounting or Sampling)

Syntax

```
interface interface-name {
    engine-id number;
    engine-type number;
    source-address address;
}
```

Hierarchy Level

[edit forwarding-options accounting *group-name* output],
[edit forwarding-options sampling family *family-name* output]

Release Information

Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description

Specify the output interface for sending copies of packets elsewhere to be analyzed.

Options

engine-id *number*—Identity of the accounting interface.

engine-type *number*—Type of this accounting interface.

interface-name—Name of the accounting interface.

source-address *address*—Address used for generating packets.

Required Privilege Level

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

Related Documentation

- Configuring Discard Accounting on page 205
- Configuring the Output File for Traffic Sampling on page 187

interface (BOOTP)

Syntax

```
interface (interface-name | interface-group) {
  broadcast;
  client-response-ttl number;
  description text-description;
  maximum-hop-count number;
  minimum-wait-time seconds;
  no-listen;
  server address {
    logical-system logical-system-name <routing-instance [ <default>
      routing-instance-names ]>;
    routing-instance [ <default> routing-instance-names ];
  }
}
```

Hierarchy Level [edit forwarding-options helpers bootp]

Release Information Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.
Statement introduced in Junos OS Release 11.3 for QFX Series switches.

Description Specify the interface for a DHCP and BOOTP relay agent.

Options ***interface-group***—Sets a logical interface or group of logical interfaces with a specific DHCP relay configuration.

The remaining statements are explained separately.

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

Related Documentation

- Configuring Routers, Switches, and Interfaces as DHCP and BOOTP Relay Agents on page 210
- Setting Up DHCP Option 82 with the Switch as a Relay Agent Between Clients and DHCP Server (CLI Procedure)

interface (DHCP Spoofing Prevention)

Syntax interface *interface-name*;

Hierarchy Level [edit routing-instances *routing-instance-name* bridge-domains *bridge-domain-name* forwarding-options dhcp-relay group *group-name*],
[edit routing-instances *routing-instance-name* forwarding-options dhcp-relay group *group-name*]

Release Information Statement introduced in Junos OS Release 9.4 (MX Series routers only).

Description Configure Dynamic Host Configuration Protocol (DHCP) snooping on the router. When acting as a snooping agent, the MX Series router typically is located between the client and the DHCP relay agent. It creates filters by "snooping" DHCP messages and binding DHCP-issued IP addresses with the MAC address of the client. These filters help prevent DHCP spoofing.

DHCP snooping is configured by including the appropriate interfaces.

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

Related Documentation

- Preventing DHCP Spoofing on MX Series 3D Universal Edge Routers on page 215

interface (DNS and TFTP Packet Forwarding or Relay Agent)

Syntax

```
interface interface-name {
  broadcast;
  description text-description;
  no-listen;
  server address <logical-system logical-system-name> <routing-instance
    routing-instance-name>;
}
```

Hierarchy Level

[edit forwarding-options helpers domain],
[edit forwarding-options helpers tftp]

Release Information

Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description

Specify the interface for monitoring and forwarding DNS or TFTP requests.

Options

interface-name—Name of the interface.

The remaining statements are explained separately.

Required Privilege Level

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

Related Documentation

- Configuring DNS and TFTP Packet Forwarding on page 212

interface (Monitoring)

Syntax

```
interface interface-name {
    engine-id number;
    engine-type number;
    input-interface-index number;
    output-interface-index number;
    source-address address;
}
```

Hierarchy Level [edit forwarding-options monitoring *group-name* inet output]

Release Information Statement introduced before Junos OS Release 7.4.

Description Specify the output interface for monitored traffic.

Options

interface-name—Name of the interface.

engine-id *number*—Identity of the monitoring interface.

engine-type *number*—Type of this monitoring interface.

input-interface-index *number*—Input interface index for records from this interface.

output-interface-index *number*—Output interface index for records from this interface.

source-address *address*—Address used for generating packets.

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

Related Documentation

- Configuring Flow Monitoring on page 206

interface (Next-Hop Group)

Syntax

```
interface interface-name {
  next-hop address;
}
```

Hierarchy Level [edit forwarding-options next-hop-group *group-name*]

Release Information Statement introduced before Junos OS Release 7.4.

Description Specify the output interface for sending copies of packets elsewhere to be analyzed.

Options ***interface-name***—Name of the interface.

The remaining statements are explained separately.

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

Related Documentation

- Configuring Next-Hop Groups on page 208

interface (Port Mirroring)

Syntax

```
interface interface-name {
  next-hop address;
}
```

Hierarchy Level [edit forwarding-options port-mirroring output],
[edit forwarding-options port-mirroring family (inet | inet6) output]

Release Information Statement introduced before Junos OS Release 7.4.

Description Specify the output interface for sending copies of packets elsewhere to be analyzed.

Options ***interface-name***—Name of the interface.

The remaining statements are explained separately.

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

Related Documentation

- Configuring Port Mirroring on page 217

load-balance

Syntax

```
load-balance {
  indexed-load-balance;
  per-flow {
    hash-seed;
  }
  per-prefix {
    hash-seed number;
  }
}
```

Hierarchy Level

[edit forwarding-options],
[edit logical-systems *logical-system-name* forwarding-options],
[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* forwarding-options],
[edit routing-instances *routing-instance-name* forwarding-options]

Release Information

Statement introduced in Junos OS Release 9.0.
Statement introduced in Junos OS Release 9.0 for EX Series switches.
Support for per-flow load balancing introduced in Junos OS Release 9.3.

Description

Enable per-prefix or per-flow load balancing so that the router or switch elects a next hop independently of the route selected by other routers or switches.

For the active route, when there are multiple equal-cost paths to the same destination, by default, the Junos OS chooses in a random fashion one of the next-hop addresses to install into the forwarding table. Whenever the set of next hops for a destination changes in any way, the next-hop address is chosen again, also in a random fashion.

You can configure the Junos OS so that, for the active route, all next-hop addresses for a destination are installed in the forwarding table. This is called per-packet load balancing. You can use load balancing to spread traffic across multiple paths between routing devices. The behavior of the per-packet load-balancing function varies according to the version of the Internet Processor ASIC in the routing device.

On routing devices with an Internet Processor I ASIC, when per-packet load balancing is configured, traffic between routing devices with multiple paths is spread in a random fashion across the available interfaces. The forwarding table balances the traffic headed to a destination, transmitting packets in round-robin fashion among the multiple next hops (up to a maximum of eight equal-cost load-balanced paths). The traffic is load-balanced on a per-packet basis.

Per-packet load distribution uses a hashing algorithm that distributes packets over equal-cost links. The algorithm is designed to distribute packets to prevent any single link from being saturated. However, per-packet load balancing offers no guarantee of equal distribution of traffic over equal-cost links, nor does it guarantee that increasing the number of Internet flows creates a better hash distribution.

On routing devices with the Internet Processor II ASIC and T Series Internet Processor II ASIC, when per-packet load balancing is configured, traffic between routing devices with multiple paths is divided into individual traffic flows (up to a maximum of 16 equal-cost

load-balanced paths). Packets for each individual flow are kept on a single interface. To recognize individual flows in the transit traffic, the routing device examines each of the following:

- Source IP address
- Destination IP address
- Protocol
- Source port number
- Destination port number
- Source interface index
- Type of service (ToS)

The routing device recognizes packets in which all of these parameters are identical, and it ensures that these packets are sent out through the same interface. This prevents problems that might otherwise occur with packets arriving at their destination out of their original sequence.

Load balancing is not supported on management and internal Ethernet (**fxo**) interfaces because this type of interface cannot handle the routing process. On **fxp** interfaces, you cannot configure multiple next hops and enable load balancing.

Options The statements are explained separately.

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

Related Documentation
- Example: Load Balancing BGP Traffic
- Configuring Per-Flow Load Balancing Based on Hash Values on page 210
- Configuring Per-Prefix Load Balancing on page 209

load-balance-group

Syntax

```
load-balance-group group-name {
  next-hop-group [ group-names ];
}
```

Hierarchy Level [edit firewall]

Release Information Statement introduced before Junos OS Release 7.4.

Description Configure a load-balance group.

Options ***group-name***—Name of load-balance group.

group-names—Name of next-hop groups to include in the load-balance group set.

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

Related Documentation

- Configuring Load-Balance Groups on page 208 in the *Junos OS Policy Framework Configuration Guide*

local-dump

Syntax (local-dump | no-local-dump);

Hierarchy Level [edit forwarding-options sampling family (inet | inet6 | mpls) output flow-server *hostname*]

Release Information Statement introduced before Junos OS Release 7.4.

Description Enable collection of cflowd records in a log file.

Options **no-local-dump**—Do not dump cflowd records to a log file before exporting.

local-dump—Dump cflowd records to a log file before exporting.

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

Related Documentation

- Debugging cflowd Flow Aggregation on page 191

max-packets-per-second

Syntax max-packets-per-second *number*;

Hierarchy Level [edit forwarding-options sampling input]

Release Information Statement introduced before Junos OS Release 7.4.

Description Specify that the traffic threshold that must be exceeded before packets are dropped. A value of 0 instructs the Packet Forwarding Engine not to sample any traffic.

Options *number*—Maximum number of packets per second.
Range: 0 through 65,535
Default: 1000

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

Related Documentation
- See Configuring Traffic Sampling on page 185.

maximum-capture-size

Syntax maximum-capture-size *bytes*;

Hierarchy Level [edit forwarding-options packet-capture]

Release Information Statement introduced in Junos OS Release 7.5.

Description Configure the maximum size of capture for packets.

Options *bytes*—Maximum capture size.
Range: 68 through 1500
Default: 68 bytes

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

Related Documentation
- Configuring Packet Capture on page 221

maximum-hop-count

Syntax maximum-hop-count *number*;

Hierarchy Level [edit forwarding-options helpers bootp],
[edit forwarding-options helpers bootpinterface (*interface-name* | *interface-group*)]

Release Information Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.
Statement introduced in Junos OS Release 11.3 for QFX Series switches.

Description Specify the maximum number of hops allowed.

Options ***number***—Maximum number of hops.

Default: 4 hops

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

Related Documentation

- Configuring Routers, Switches, and Interfaces as DHCP and BOOTP Relay Agents on page 210

maximum-packet-length

Syntax maximum-packet-length *bytes*;

Hierarchy Level [edit forwarding-options port-mirroring input],
[edit forwarding-options port-mirroring instance *instance-name* input],
[edit forwarding-options sampling input],
[edit forwarding-options sampling instance *instance-name* input]

Release Information Statement introduced in Junos OS Release 9.6.

Description Set the maximum length of the packet used for port mirroring or traffic sampling. Packets with lengths greater than the specified maximum are truncated.

NOTE: The **maximum-packet-length** statement is not supported on MX80 routers.

Options ***bytes***—Number of bytes.

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

Related Documentation

- Configuring Port Mirroring on page 217
- Configuring Traffic Sampling on page 185

minimum-wait-time

Syntax minimum-wait-time *seconds*;

Hierarchy Level [edit forwarding-options helpers bootp],
[edit forwarding-options helpers bootpinterface (*interface-name* | *interface-group*)]

Release Information Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.
Statement introduced in Junos OS Release 11.3 for QFX Series switches.

Description Specify the minimum time allowed.

Options ***seconds***—Minimum time.

Default: 0 seconds

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

Related Documentation
- Configuring Routers, Switches, and Interfaces as DHCP and BOOTP Relay Agents on page 210

mirror-once

Syntax mirror-once;

Hierarchy Level [edit forwarding-options port-mirroring]

Release Information Statement introduced in Junos OS Release 9.3 (MX Series routers only).
Support extended to M120 routers in Junos OS Release 9.5.

Description Configure the router to mirror packets only once. This feature is useful if you configure port mirroring on both ingress and egress interfaces, which could result in the same packet being mirrored twice.

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

Related Documentation
- Configuring Port Mirroring on page 217

monitoring

Syntax

```
monitoring group-name {
  family inet {
    output {
      cflowd hostname {
        port port-number;
      }
      export-format cflowd-version-5;
      flow-active-timeout seconds;
      flow-export-destination {
        (cflowd-collector | collector-pic);
      }
      flow-inactive-timeout seconds;
      interface interface-name {
        engine-id number;
        engine-type number;
        input-interface-index number;
        output-interface-index number;
        source-address address;
      }
    }
  }
}
```

Hierarchy Level [edit forwarding-options]

Release Information Statement introduced before Junos OS Release 7.4.

Description Specify flow monitoring instance name and properties.

The statements are explained separately.

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

Related Documentation

- Configuring Flow Monitoring on page 206

next-hop

Syntax next-hop *address*;

Hierarchy Level [edit forwarding-options port-mirroringoutput interface *interface-name*],
[edit forwarding-options port-mirroring family (inet | inet6 | ccc | vpls) output interface *interface-name*]

Release Information Statement introduced before Junos OS Release 7.4.

Description Specify the next-hop address for sending copies of packets to an analyzer.

Options ***address***—IP address of the next-hop router.

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

Related Documentation

- Configuring Port Mirroring on page 217

next-hop-group

Syntax

```
next-hop-group group-name {
  interface interface-name {
    next-hop address;
  }
  next-hop-subgroup subgroup-name {
    interface interface-name {
      next-hop address;
    }
  }
}
```

Hierarchy Level [edit forwarding-options]

Release Information Statement introduced before Junos OS Release 7.4.

Description Specify the next-hop address for sending copies of packets to an analyzer.

Options

addresses—IP address of the next-hop router. Each next-hop group supports up to 16 next-hop addresses. Up to 30 next-hop groups are supported. Each next-hop group must have at least two next-hop addresses.

group-names—Name of next-hop group. Up to 30 next-hop groups are supported for the router. Each next-hop group must have at least two next-hop addresses.

interface-name—Interface used to reach the next-hop destination.

Required Privilege Level

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

Related Documentation

- Configuring Next-Hop Groups on page 208

no-filter-check

Syntax no-filter-check;

Hierarchy Level [edit forwarding-options port-mirroring output],
[edit forwarding-options port-mirroring family (inet | inet6 | ccc | vpls) output]

Release Information Statement introduced before Junos OS Release 7.4.

Description Disable filter checking on the port-mirroring interface.

This statement is required when you send port-mirrored traffic to a Tunnel Services PIC that has a filter applied to it.

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

Related Documentation
- Configuring Port Mirroring on page 217

no-listen

Syntax no-listen;

Hierarchy Level [edit forwarding-options helpers bootp interface (*interface-name* | *interface-group*)],
[edit forwarding-options helpers domain interface *interface-name*],
[edit forwarding-options helpers tftp interface *interface-name*]

Release Information Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.
Statement introduced in Junos OS Release 11.3 for QFX Series switches.

Description Disable recognition of DNS requests or stop packets from being forwarded on a logical interface, a group of logical interfaces, a router, or a switch.

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

Related Documentation
- Configuring DNS and TFTP Packet Forwarding on page 212
- Configuring Routers, Switches, and Interfaces as DHCP and BOOTP Relay Agents on page 210

output (Accounting)

Syntax

```
output {
  cflowd [ hostnames ] {
    aggregation {
      autonomous-system;
      destination-prefix;
      protocol-port;
      source-destination-prefix {
        caida-compliant;
      }
      source-prefix;
    }
    autonomous-system-type (origin | peer);
    port port-number;
    version format;
  }
  flow-active-timeout seconds;
  flow-inactive-timeout seconds;
  interface interface-name {
    engine-id number;
    engine-type number;
    source-address address;
  }
}
```

Hierarchy Level [edit forwarding-options accounting *group-name*]

Release Information Statement introduced before Junos OS Release 7.4.

Description Configure cflowd, output interfaces, and flow properties.

The statements are explained separately.

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

Related Documentation

- Configuring Discard Accounting on page 205

output (Forwarding Table)

Syntax output *filter-name*;

Hierarchy Level [edit forwarding-optionsfamily (inet | inet6 | mpls) filter],
[edit routing-instances *routing-instance-name* forwarding-optionsfamily (inet | inet6 | mpls) filter]

Release Information Statement introduced in Junos OS Release 7.5.

Description Configure filtering on the egress traffic of the forwarding table.

Options ***filter-name***—Name of the applied filter.

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

Related Documentation

- Applying Filters to Forwarding Tables on page 203

output (Monitoring)

Syntax

```
output {
  cflowd hostname {
    port port-number;
  }
  export-format cflowd-version-5;
  flow-active-timeout seconds;
  flow-export-destination {
    (cflowd-collector | collector-pic);
  }
  flow-inactive-timeout seconds;
  interface interface-name {
    engine-id number;
    engine-type number;
    input-interface-index number;
    output-interface-index number;
    source-address address;
  }
}
```

Hierarchy Level [edit forwarding-options monitoring *group-name* family inet]

Release Information Statement introduced before Junos OS Release 7.4.

Description Configure cflowd, output interfaces, and flow properties.

The statements are explained separately.

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

Related Documentation

- Configuring Flow Monitoring on page 206

output (Port Mirroring)

Syntax

```
output {
  interface interface-name {
    next-hop address;
  }
  no-filter-check;
}
```

Hierarchy Level

[edit forwarding-options port-mirroring family (ccc | inet | inet6 | vpls)],
[edit forwarding-options port-mirroring instance *instance-name* family (ccc | inet | inet6 | vpls)]

Release Information

Statement introduced before Junos OS Release 7.4.
vpls option introduced in Junos OS Release 9.3 for MX Series routers only; support extended to M7i, M10i, M120, and M320 routers in Junos OS Release 9.5.
ccc option introduced in Junos OS Release 9.6 for M120 and M320 routers only.

Description

Configure the port mirroring destination properties.

The statements are explained separately.

Required Privilege Level

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

Related Documentation

- Configuring Port Mirroring on page 217

output (Sampling)

Syntax

```
output {
  aggregate-export-interval seconds;
  flow-server hostname {
    aggregation {
      autonomous-system;
      destination-prefix;
      protocol-port;
      source-destination-prefix {
        caida-compliant;
      }
      source-prefix;
    }
    autonomous-system-type (origin | peer);
    (local-dump | no-local-dump);
    port port-number;
    source-address address;
    version format;
    version9 {
      template template-name;
    }
  }
  extension-service service-name;
  file filename filename <disable> <files number> <stamp | no-stamp> <size bytes>
    <world-readable | no-world-readable>;
  flow-active-timeout seconds;
  flow-inactive-timeout seconds;
  flow-server host-name {
    aggregation;
    autonomous-system-type (origin | peer);
    (local-dump | no-local-dump);
    port number;
    source-address address;
    version (5 | 8);
    version9;
  }
  interface interface-name {
    engine-id number;
    engine-type number;
    source-address address;
  }
}
```

Hierarchy Level [edit forwarding-options sampling family (inet | inet6 | mpls)]

Release Information Statement introduced before Junos OS Release 7.4.
version9 statement introduced in Junos OS Release 8.3.

Description Configure cflowd, output files and interfaces, and flow properties. Enable the collection of traffic flows using the version 9 format.

The statements are explained separately.

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

Related Documentation
- Configuring Active Flow Monitoring Using Version 9 on page 192
- Configuring the Output File for Traffic Sampling on page 187

packet-capture

Syntax

```
packet-capture {
  disable;
  filefilename filename <files number> <size bytes> <world-readable | no-world-readable>;
  maximum-capture-size number;
}
```

Hierarchy Level
[edit forwarding-options]

Release Information
Statement introduced in Junos OS Release 7.5.

Description
Configure packet capture on a router.

Options
disable—Disable packet capture on the router.

The remaining statements are explained separately.

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

Related Documentation
- Configuring Packet Capture on page 221

per-flow

Syntax

```
per-flow {
  hash-seed;
}
```

Hierarchy Level

[edit forwarding-options load-balance],
[edit logical-systems *logical-system-name* forwarding-options load-balance],
[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* forwarding-options load-balance],
[edit routing-instances *routing-instance-name* forwarding-options load-balance]

Release Information

Statement introduced in Junos OS Release 9.3 (M120, M320, and MX Series routers only).

Description

Enable per-flow load balancing based on hash values.

Options

hash-seed—Configure the hash value. The Junos OS automatically chooses a value for the hashing algorithm used. You cannot configure a specific hash value for per-flow load balancing.

Required Privilege Level

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

Related Documentation

- Configuring Per-Flow Load Balancing Based on Hash Values on page 210
- load-balance on page 298

per-prefix

Syntax

```
per-prefix {
  hash-seed number;
}
```

Hierarchy Level

[edit forwarding-options load-balance],
[edit logical-systems *logical-system-name* forwarding-options load-balance],
[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* forwarding-options load-balance],
[edit routing-instances *routing-instance-name* forwarding-options load-balance]

Release Information

Statement introduced in Junos OS Release 9.0.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description

Configure the hash parameter for per-prefix load balancing.

Options

hash-seed—Per-prefix load-balancing hash function.

number—Hash value.

Range: 0 through 65,534

Default: 0

Required Privilege Level

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

Related Documentation

- Configuring Per-Prefix Load Balancing on page 209
- load-balance on page 298

port

Syntax

port *port-number*;

Hierarchy Level

[edit forwarding-options accounting *group-name* output cflowd *hostname*],
[edit forwarding-options monitoring *group-name* family inet output flow-server *hostname*],
[edit forwarding-options sampling family (inet | inet6 | mpls) output flow-server *hostname*]

Release Information

Statement introduced before Junos OS Release 7.4.

Description

Specify the UDP port number on the cflowd host system.

Options

port-number—Any valid UDP port number on the host system.

Required Privilege Level

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

Related Documentation

- Configuring Flow Aggregation (cflowd) on page 189

port-mirroring

Syntax

```
port-mirroring {
  input {
    maximum-packet-length bytes;
    rate number;
    run-length number;
  }
  family (ccc | inet | inet6 | vpls) {
    output {
      interface interface-name {
        next-hop address;
      }
      no-filter-check;
    }
  }
  instance {
    instance-name {
      input {
        maximum-packet-length bytes;
        rate number;
        run-length number;
      }
      family (ccc | inet | inet6 | vpls) {
        output {
          interface interface-name {
            next-hop address;
          }
          no-filter-check;
        }
      }
    }
  }
  mirror-once;
  traceoptions {
    file filename <files number> <size bytes> <world-readable | no-world-readable>;
  }
}
```

Hierarchy Level [edit forwarding-options]

Release Information

Statement introduced before Junos OS Release 7.4.
family vpls statement introduced in Junos OS Release 9.3 (MX Series routers only); support extended to M7i, M10, M120, and M320 routers in Junos OS Release 9.5.
instance ***port-mirroring-instance-name*** statement introduced in Junos OS Release 9.3 (MX Series routers only); support extended to M120 and M320 routers in Junos OS Release 9.5.
mirror-once statement introduced in Junos OS Release 9.3 (MX Series routers only); support extended to M120 routers in Junos OS Release 9.5.
family ccc statement introduced in Junos OS Release 9.6 (M120 and M320 routers only)

Description Specify the address family, rate, run length, interface, and next-hop address for sending copies of packets to an analyzer.

The statements are explained separately.

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

Related Documentation
- Configuring Port Mirroring on page 217
- *Junos OS MX Series 3D Universal Edge Routers Solutions Guide*

rate

Syntax
rate *number*;

Hierarchy Level
[edit forwarding-options port-mirroring input],
[edit forwarding-options port-mirroring instance *instance-name* input],
[edit forwarding-options samplinginput]

Release Information
Statement introduced before Junos OS Release 7.4.

Description
Set the ratio of the number of packets to be sampled. For example, if you specify a rate of 10, every tenth packet (1 packet out of 10) is sampled.

Options
number—Denominator of the ratio.
Range: 1 through 65,535

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

Related Documentation
- Configuring Port Mirroring on page 217
- Configuring Traffic Sampling on page 185

relay-agent-option

Syntax relay-agent-option;

Hierarchy Level [edit logical-systems routing-instances *instance-name* forwarding-options helpers bootp],
[edit forwarding-options helpers bootp],
[edit routing-instances *instance-name* forwarding-options helpers bootp],

Release Information Statement introduced in Junos OS Release 8.0.

Description Enables the DHCP relay agent information option which allows DHCP to forward information between different VRF routing instances. The functionality is described in RFC 3046, *DHCP Relay Agent Information Option*. For the Junos OS implementation, the DHCP option number is 82 and the suboption ID is 1. The suboption length is the length required to contain an interface name in addition to the terminating null character. The overall option length is the suboption length plus 2 bytes (for the option header). The DHCP relay agent information option is only present on packets sent between the relay and the DHCP server.

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

Related Documentation
- Configuring Routers, Switches, and Interfaces as DHCP and BOOTP Relay Agents on page 210

route-accounting

Syntax route-accounting;

Hierarchy Level [edit forwarding-optionsfamily inet6]

Release Information Statement introduced in Junos OS Release 8.3.

Description Configure the routing platform to track IPv6 traffic passing through the router.

Default Disabled

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

Related Documentation
- Configuring IPv6 Accounting on page 205

run-length

Syntax run-length *number*;

Hierarchy Level [edit forwarding-options port-mirroringinput],
[edit forwarding-options port-mirroring instance *port-mirroring-instance-name* input],
[edit forwarding-options samplinginput]
[edit forwarding-options sampling instance *instance-name* input]

Release Information Statement introduced before Junos OS Release 7.4.

Description Set the number of samples following the initial trigger event. This allows you to sample packets following those already being sampled.

NOTE: The run-length statement is not supported on MX80 routers.

Options *number*—Number of samples.

Range: 0 through 20

Default: 0

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

Related Documentation

- Applying Filters to Forwarding Tables on page 203

sampling

Syntax

```
sampling {
    disable;
    input {
        max-packets-per-second number;
        maximum-packet-length bytes;
        rate number;
        run-length number;
    }
    family (inet | inet6 | mpls) {
        output {
            aggregate-export-interval seconds;
            flow-server hostname {
                aggregation {
                    autonomous-system;
                    destination-prefix;
                    protocol-port;
                    source-destination-prefix {
                        caida-compliant;
                    }
                    source-prefix;
                }
                autonomous-system-type (origin | peer);
                (local-dump | no-local-dump);
                port port-number;
                source-address address;
                version format;
                version9 {
                    template template-name;
                }
            }
        }
        extension-service service-name;
        file filename filename <disable> <files number> <stamp | no-stamp> <size bytes>
            <world-readable | no-world-readable>;
        flow-active-timeout seconds;
        flow-inactive-timeout seconds;
        flow-server host-name {
            aggregation;
            autonomous-system-type (origin | peer);
            (local-dump | no-local-dump);
            port number;
            source-address address;
            version (5 | 8);
            version9;
        }
        input interface-name {
            engine-id number;
            engine-type number;
            source-address address;
        }
    }
    traceoptions {
```

```
    file filename <files number> < bytes> <world-readable | no-world-readable>;
  }
}
```

Hierarchy Level [edit forwarding-options]

Release Information Statement introduced before Junos OS Release 7.4.

Description Configure traffic sampling.

The statements are explained separately.

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

Related Documentation

- Applying Filters to Forwarding Tables on page 203
- Configuring Active Flow Monitoring Using Version 9 on page 192
- Configuring Flow Aggregation (cflowd) on page 189
- Configuring Port Mirroring on page 217
- Tracing Traffic-Sampling Operations on page 189

server (DHCP and BOOTP Relay Agent)

Syntax

```
server address {
  logical-system logical-system-name <routing-instance [ <default>
    routing-instance-names ]>;
  routing-instance [ <default> routing-instance-names ];
}
```

Hierarchy Level [edit forwarding-options helpers bootp],
[edit forwarding-options helpers bootp interface (*interface-name* | *interface-group*)]

Release Information Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.
Statement introduced in Junos OS Release 11.3 for QFX Series switches.

Description Configure the router or switch to act as a DHCP and BOOTP relay agent.

Options

- ***address***—One or more addresses of the server.
- **logical-system *logical-system-name***—(Optional) Logical system of the server.
- **routing-instance *routing-instance-names***—(Optional) Routing instance name that belong to the DHCP or BOOTP relay agent.

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

Related Documentation

- Configuring Routers, Switches, and Interfaces as DHCP and BOOTP Relay Agents on page 210

server (DNS and TFTP Service)

Syntax server *address* <logical-system *logical-system-name*> <routing-instance *routing-instance-name*>;

Hierarchy Level [edit forwarding-options helpers domain],
[edit forwarding-options helpers domain interface *interface-name*],
[edit forwarding-options helpers tftp],
[edit forwarding-options helpers tftp interface *interface-name*]

Release Information Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Specify the DNS or TFTP server for forwarding DNS or TFTP requests. Only one server can be specified for each interface.

Options ***address***—Address of the server.

logical-system *logical-system-name*—(Optional) Logical system of the server.

routing-instance [*routing-instance-names*]—(Optional) Set the routing instance name or names that belong to the DNS server or TFTP server.

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

Related Documentation
- Configuring DNS and TFTP Packet Forwarding on page 212

size (Packet Capture)

Syntax size *number*;

Hierarchy Level [edit forwarding-options packet-capture file]

Release Information Statement introduced in Junos OS Release 7.5.

Description Configure the maximum size of the file for packet capturing.

Options ***number***—Maximum size of file.
Range: 1024 through 104,857,600 bytes
Default: 512,000 bytes

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

Related Documentation
- Configuring Packet Capture on page 221

size (Sampling and Traceoptions)

Syntax size *bytes*;

Hierarchy Level [edit forwarding-options helperstraceoptions file],
[edit forwarding-options port-mirroring traceoptions file],
[edit forwarding-options sampling family *family-name* output file],
[edit forwarding-options sampling traceoptions file]

Release Information Statement introduced before Junos OS Release 7.4.

Description Specify the maximum size of each file containing sample or log data. The file size is limited by the number of files to be created and the available hard disk space.

When a traffic sampling file named **sampling-file** reaches the maximum size, it is renamed **sampling-file.0**. When the **sampling-file** file again reaches its maximum size, **sampling-file.0** is renamed **sampling-file.1** and **sampling-file** is renamed **sampling-file.0**. This renaming scheme continues until the maximum number of traffic sampling files is reached. Then the oldest traffic sampling file is overwritten.

Options ***bytes***—Maximum size of each traffic sampling file or trace log file, in kilobytes (KB), megabytes (MB), or gigabytes (GB).

Syntax: ***x*k** to specify KB, ***x*m** to specify MB, or ***x*g** to specify GB

Range: 10 KB through the maximum file size supported on your router

Default: 1 MB for sampling data; 128 KB for log information

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

Related Documentation

- Configuring the Output File for Traffic Sampling on page 187
- Tracing Traffic-Sampling Operations on page 189

stamp

Syntax (stamp | no-stamp);

Hierarchy Level [edit forwarding-options sampling family *family-name* output file]

Release Information Statement introduced before Junos OS Release 7.4.

Description Include a timestamp with each line in the output file.

Default **no-stamp**

Options **no-stamp**—Do not include timestamps.

stamp—Include a timestamp with each line of packet sampling information.

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

Related Documentation
- Configuring the Output File for Traffic Sampling on page 187

tftp

Syntax

```
tftp {
  description text-description;
  interface interface-name {
    broadcast;
    description text-description;
    no-listen;
    server address <logical-system logical-system-name> <routing-instance
      routing-instance-name>;
  }
  server address <logical-system logical-system-name> <routing-instance
    routing-instance-name>;
}
```

Hierarchy Level [edit forwarding-options helpers]

Release Information Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Enable TFTP request packet forwarding.

The remaining statements are explained separately.

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

Related Documentation
- Configuring DNS and TFTP Packet Forwarding on page 212

traceoptions (DNS and TFTP Packet Forwarding)

Syntax

```
traceoptions {
  file filename <files number> <match regular-expression> <size bytes> <world-readable |
    no-world-readable>;
  flag flag;
  level level;
  <no-remote-trace>;
}
```

Hierarchy Level [edit forwarding-options helpers]

Release Information Statement introduced before Junos OS Release 7.4.
Statement standardized and **match** option introduced in Junos OS Release 8.0.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Configure tracing operations for BOOTP, DNS and TFTP packet forwarding.

Default If you do not include this statement, no tracing operations are performed.

Options

file *filename*—Name of the file to receive the output of the tracing operation. Enclose the name in quotation marks (" "). All files are placed in a file named **fud** in the directory **/var/log**. If you include the **file** statement, you must specify a filename.

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

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

Range: 2 through 1000

Default: 3 files

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

- **address**—Trace address management events
- **all**—Trace all events
- **bootp**—Trace BOOTP or DHCP services events
- **config**—Trace configuration events
- **domain**—Trace DNS service events
- **ifdb**—Trace interface database operations
- **io**—Trace I/O operations
- **main**—Trace main loop events
- **port**—Trace arbitrary protocol events

- **rtsock**—Trace routing socket operations
- **tftp**—Trace TFTP service events
- **trace**—Trace tracing operations
- **ui**—Trace user interface operations
- **util**—Trace miscellaneous utility operations

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

no-remote-trace—(Optional) Disable remote tracing globally or for a specific tracing operation.

no-world-readable—(Optional) Restrict file access to the owner.

size *size*—(Optional) Maximum size of each trace file, in kilobytes (KB), megabytes (MB), or gigabytes (GB). When a trace file named **trace-file** reaches this size, it is renamed **trace-file.0**. When the **trace-file** file again reaches its maximum size, **trace-file.0** is renamed **trace-file.1** and **trace-file** is renamed **trace-file.0**. This renaming scheme continues until the maximum number of trace files is reached. Then the oldest trace file is overwritten.

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

Syntax: ***x*k** to specify KB, ***x*m** to specify MB, or ***x*g** to specify GB

Range: 0 bytes through 4,294,967,295 KB

Default: 128 KB

world-readable—(Optional) Enable unrestricted file access.

Required Privilege Level

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

Related Documentation

- Tracing BOOTP, DNS, and TFTP Forwarding Operations on page 213

traceoptions (Port Mirroring and Traffic Sampling)

Syntax

```
traceoptions {
    file filename <files number> <size bytes> <world-readable | no-world-readable>;
}
```

Hierarchy Level

[edit forwarding-options port-mirroring],
[edit forwarding-options sampling]

Release Information

Statement introduced before Junos OS Release 7.4.

Description

Configure traffic sampling tracing operations.

The statements are explained separately.

Required Privilege Level

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

Related Documentation

- Tracing Traffic-Sampling Operations on page 189

version

Syntax

version *format*;

Hierarchy Level

[edit forwarding-options accounting *group-name* output cflowd *hostname*],
[edit forwarding-options sampling family (inet | inet6 | mpls) output flow-server *hostname*]

Release Information

Statement introduced before Junos OS Release 7.4.

Description

Specify the version format of the aggregated flows exported to a cflowd server.

Options

format—Export format of the flows.

Values: 5 or 8

Default: 5

Required Privilege Level

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

Related Documentation

- Configuring Flow Aggregation (cflowd) on page 189

version9

Syntax

```
version9 {
  template template-name;
}
```

Hierarchy Level [edit forwarding-options sampling family *family-name* output flow-server *hostname*]

Release Information Statement introduced in Junos OS Release 8.3.

Description Enable active flow monitoring using the version 9 template format to collect traffic flows.

Options **template *template-name***—Name of a version 9 record flow format template configured at the **[edit services monitoring]** hierarchy level.

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

Related Documentation

- Configuring Active Flow Monitoring Using Version 9 on page 192
- *Junos OS Flow Monitoring Feature Guide*
- *Junos OS Services Interfaces Configuration Guide*

world-readable

Syntax (world-readable | no-world-readable);

Hierarchy Level [edit forwarding-options helperstraceoptions file],
[edit forwarding-options packet-capture file],
[edit forwarding-options port-mirroring traceoptions file],
[edit forwarding-options sampling family *family-name* output file],
[edit forwarding-options samplingtraceoptions file]

Release Information Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Enable unrestricted file access.

Default **no-world-readable**

Options **no-world-readable**—Restrict file access to the owner.

world-readable—Enable unrestricted file access.

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

Related Documentation

- Configuring the Output File for Traffic Sampling on page 187
- Tracing Traffic-Sampling Operations on page 189

PART 4

Index

Index

Symbols

A

B

C

D

E

F

J

L

M

N

O

P

R

S

T

U

V

W