



JUNOS® Software

Policy Framework Configuration Guide

Release 10.0

Juniper Networks, Inc.

1194 North Mathilda Avenue
Sunnyvale, California 94089
USA

408-745-2000

www.juniper.net

Published: 2009-10-13

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

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

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

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

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

Juniper Networks, the Juniper Networks logo, JUNOS, NetScreen, ScreenOS, and Steel-Belted Radius are registered trademarks of Juniper Networks, Inc. in the United States and other countries. JUNOSe is a trademark of Juniper Networks, Inc. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners.

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

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

JUNOS® Software Policy Framework Configuration Guide.

Release 10.0

Copyright © 2009, Juniper Networks, Inc.

All rights reserved. Printed in USA.

Writing: Ines Salazar, Fran Singer, Alan Twigg, Lisa Kelly

Editing: Nancy Kurahashi

Illustration: Faith Bradford, Nathaniel Woodward

Cover Design: Edmonds Design

Revision History

October 2009—JUNOS 10.0R1

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

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. The JUNOS Software 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

READ THIS END USER LICENSE AGREEMENT ("AGREEMENT") BEFORE DOWNLOADING, INSTALLING, OR USING THE SOFTWARE. BY DOWNLOADING, INSTALLING, OR USING THE SOFTWARE OR OTHERWISE EXPRESSING YOUR AGREEMENT TO THE TERMS CONTAINED HEREIN, YOU (AS CUSTOMER OR IF YOU ARE NOT THE CUSTOMER, AS A REPRESENTATIVE/AGENT AUTHORIZED TO BIND THE CUSTOMER) CONSENT TO BE BOUND BY THIS AGREEMENT. IF YOU DO NOT OR CANNOT AGREE TO THE TERMS CONTAINED HEREIN, THEN (A) DO NOT DOWNLOAD, INSTALL, OR USE THE SOFTWARE, AND (B) YOU MAY CONTACT JUNIPER NETWORKS REGARDING LICENSE TERMS.

1. **The Parties.** The parties to this Agreement are (i) Juniper Networks, Inc. (if the Customer's principal office is located in the Americas) or Juniper Networks (Cayman) Limited (if the Customer's principal office is located outside the Americas) (such applicable entity being referred to herein as "Juniper"), and (ii) the person or organization that originally purchased from Juniper or an authorized Juniper reseller the applicable license(s) for use of the Software ("Customer") (collectively, the "Parties").

2. **The Software.** In this Agreement, "Software" means the program modules and features of the Juniper or Juniper-supplied software, for which Customer has paid the applicable license or support fees to Juniper or an authorized Juniper reseller, or which was embedded by Juniper in equipment which Customer purchased from Juniper or an authorized Juniper reseller. "Software" also includes updates, upgrades and new releases of such software. "Embedded Software" means Software which Juniper has embedded in or loaded onto the Juniper equipment and any updates, upgrades, additions or replacements which are subsequently embedded in or loaded onto the equipment.

3. **License Grant.** Subject to payment of the applicable fees and the limitations and restrictions set forth herein, Juniper grants to Customer a non-exclusive and non-transferable license, without right to sublicense, to use the Software, in executable form only, subject to the following use restrictions:

- a. Customer shall use Embedded Software solely as embedded in, and for execution on, Juniper equipment originally purchased by Customer from Juniper or an authorized Juniper reseller.
- b. Customer shall use the Software on a single hardware chassis having a single processing unit, or as many chassis or processing units for which Customer has paid the applicable license fees; provided, however, with respect to the Steel-Belted Radius or Odyssey Access Client software only, Customer shall use such Software on a single computer containing a single physical random access memory space and containing any number of processors. Use of the Steel-Belted Radius or IMS AAA software on multiple computers or virtual machines (e.g., Solaris zones) requires multiple licenses, regardless of whether such computers or virtualizations are physically contained on a single chassis.
- c. Product purchase documents, paper or electronic user documentation, and/or the particular licenses purchased by Customer may specify limits to Customer's use of the Software. Such limits may restrict use to a maximum number of seats, registered endpoints, concurrent users, sessions, calls, connections, subscribers, clusters, nodes, realms, devices, links, ports or transactions, or require the purchase of separate licenses to use particular features, functionalities, services, applications, operations, or capabilities, or provide throughput, performance, configuration, bandwidth, interface, processing, temporal, or geographical limits. In addition, such limits may restrict the use of the Software to managing certain kinds of networks or require the Software to be used only in conjunction with other specific Software. Customer's use of the Software shall be subject to all such limitations and purchase of all applicable licenses.
- d. For any trial copy of the Software, Customer's right to use the Software expires 30 days after download, installation or use of the Software. Customer may operate the Software after the 30-day trial period only if Customer pays for a license to do so. Customer may not extend or create an additional trial period by re-installing the Software after the 30-day trial period.
- e. The Global Enterprise Edition of the Steel-Belted Radius software may be used by Customer only to manage access to Customer's enterprise network. Specifically, service provider customers are expressly prohibited from using the Global Enterprise Edition of the Steel-Belted Radius software to support any commercial network access services.

The foregoing license is not transferable or assignable by Customer. No license is granted herein to any user who did not originally purchase the applicable license(s) for the Software from Juniper or an authorized Juniper reseller.

4. **Use Prohibitions.** Notwithstanding the foregoing, the license provided herein does not permit the Customer to, and Customer agrees not to and shall not: (a) modify, unbundle, reverse engineer, or create derivative works based on the Software; (b) make unauthorized copies of the Software (except as necessary for backup purposes); (c) rent, sell, transfer, or grant any rights in and to any copy of the Software, in any form, to any third party; (d) remove any proprietary notices, labels, or marks on or in any copy of the Software or any product in which the Software is embedded; (e) distribute any copy of the Software to any third party, including as may be embedded in Juniper equipment sold in the secondhand market; (f) use any 'locked' or key-restricted feature, function, service, application, operation, or capability without first purchasing the applicable license(s) and obtaining a valid key from Juniper, even if such feature, function, service, application, operation, or capability is enabled without a key; (g) distribute any key for the Software provided by Juniper to any third party; (h) use the Software in any manner that extends or is broader than the uses purchased by Customer from Juniper or an authorized Juniper reseller; (i) use Embedded Software on non-Juniper equipment; (j) use Embedded Software (or make it available for use) on Juniper equipment that the Customer did not originally purchase from Juniper or an authorized Juniper reseller; (k) disclose the results of testing or benchmarking of the Software to any third party without the prior written consent of Juniper; or (l) use the Software in any manner other than as expressly provided herein.

5. **Audit.** Customer shall maintain accurate records as necessary to verify compliance with this Agreement. Upon request by Juniper, Customer shall furnish such records to Juniper and certify its compliance with this Agreement.

6. **Confidentiality.** The Parties agree that aspects of the Software and associated documentation are the confidential property of Juniper. As such, Customer shall exercise all reasonable commercial efforts to maintain the Software and associated documentation in confidence, which at a minimum includes restricting access to the Software to Customer employees and contractors having a need to use the Software for Customer's internal business purposes.

7. **Ownership.** Juniper and Juniper's licensors, respectively, retain ownership of all right, title, and interest (including copyright) in and to the Software, associated documentation, and all copies of the Software. Nothing in this Agreement constitutes a transfer or conveyance of any right, title, or interest in the Software or associated documentation, or a sale of the Software, associated documentation, or copies of the Software.

8. **Warranty, Limitation of Liability, Disclaimer of Warranty.** The warranty applicable to the Software shall be as set forth in the warranty statement that accompanies the Software (the "Warranty Statement"). Nothing in this Agreement shall give rise to any obligation to support the Software. Support services may be purchased separately. Any such support shall be governed by a separate, written support services agreement. TO THE MAXIMUM EXTENT PERMITTED BY LAW, JUNIPER SHALL NOT BE LIABLE FOR ANY LOST PROFITS, LOSS OF DATA, OR COSTS OR PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, OR FOR ANY SPECIAL, INDIRECT, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS AGREEMENT, THE SOFTWARE, OR ANY JUNIPER OR JUNIPER-SUPPLIED SOFTWARE. IN NO EVENT SHALL JUNIPER BE LIABLE FOR DAMAGES ARISING FROM UNAUTHORIZED OR IMPROPER USE OF ANY JUNIPER OR JUNIPER-SUPPLIED SOFTWARE, EXCEPT AS EXPRESSLY PROVIDED IN THE WARRANTY STATEMENT TO THE EXTENT PERMITTED BY LAW, JUNIPER DISCLAIMS ANY AND ALL WARRANTIES IN AND TO THE SOFTWARE (WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE), INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT DOES JUNIPER WARRANT THAT THE SOFTWARE, OR ANY EQUIPMENT OR NETWORK RUNNING THE SOFTWARE, WILL OPERATE WITHOUT ERROR OR INTERRUPTION, OR WILL BE FREE OF VULNERABILITY TO INTRUSION OR ATTACK. In no event shall Juniper's or its suppliers' or licensors' liability to Customer, whether in contract, tort (including negligence), breach of warranty, or otherwise, exceed the price paid by Customer for the Software that gave rise to the claim, or if the Software is embedded in another Juniper product, the price paid by Customer for such other product. Customer acknowledges and agrees that Juniper has set its prices and entered into this Agreement in reliance upon the disclaimers of warranty and the limitations of liability set forth herein, that the same reflect an allocation of risk between the Parties (including the risk that a contract remedy may fail of its essential purpose and cause consequential loss), and that the same form an essential basis of the bargain between the Parties.

9. **Termination.** Any breach of this Agreement or failure by Customer to pay any applicable fees due shall result in automatic termination of the license granted herein. Upon such termination, Customer shall destroy or return to Juniper all copies of the Software and related documentation in Customer's possession or control.

10. **Taxes.** All license fees payable under this agreement are exclusive of tax. Customer shall be responsible for paying Taxes arising from the purchase of the license, or importation or use of the Software. If applicable, valid exemption documentation for each taxing jurisdiction shall be provided to Juniper prior to invoicing, and Customer shall promptly notify Juniper if their exemption is revoked or modified. All payments made by Customer shall be net of any applicable withholding tax. Customer will provide reasonable assistance to Juniper in connection with such withholding taxes by promptly: providing Juniper with valid tax receipts and other required documentation showing Customer's payment of any withholding taxes; completing appropriate applications that would reduce the amount of withholding tax to be paid; and notifying and assisting Juniper in any audit or tax proceeding related to transactions hereunder. Customer shall comply with all applicable tax laws and regulations, and Customer will promptly pay or reimburse Juniper for all costs and damages related to any liability incurred by Juniper as a result of Customer's non-compliance or delay with its responsibilities herein. Customer's obligations under this Section shall survive termination or expiration of this Agreement.

11. **Export.** Customer agrees to comply with all applicable export laws and restrictions and regulations of any United States and any applicable foreign agency or authority, and not to export or re-export the Software or any direct product thereof in violation of any such restrictions, laws or regulations, or without all necessary approvals. Customer shall be liable for any such violations. The version of the Software supplied to Customer may contain encryption or other capabilities restricting Customer's ability to export the Software without an export license.

12. **Commercial Computer Software.** The Software is "commercial computer software" and is provided with restricted rights. Use, duplication, or disclosure by the United States government is subject to restrictions set forth in this Agreement and as provided in DFARS 227.7201 through 227.7202-4, FAR 12.212, FAR 27.405(b)(2), FAR 52.227-19, or FAR 52.227-14(ALT III) as applicable.

13. **Interface Information.** To the extent required by applicable law, and at Customer's written request, Juniper shall provide Customer with the interface information needed to achieve interoperability between the Software and another independently created program, on payment of applicable fee, if any. Customer shall observe strict obligations of confidentiality with respect to such information and shall use such information in compliance with any applicable terms and conditions upon which Juniper makes such information available.

14. **Third Party Software.** Any licensor of Juniper whose software is embedded in the Software and any supplier of Juniper whose products or technology are embedded in (or services are accessed by) the Software shall be a third party beneficiary with respect to this Agreement, and such licensor or vendor shall have the right to enforce this Agreement in its own name as if it were Juniper. In addition, certain third party software may be provided with the Software and is subject to the accompanying license(s), if any, of its respective owner(s). To the extent portions of the Software are distributed under and subject to open source licenses obligating Juniper to make the source code for such portions publicly available (such as the GNU General Public License ("GPL") or the GNU Library General Public License ("LGPL")), Juniper will make such source code portions (including Juniper modifications, as appropriate) available upon request for a period of up to three years from the date of distribution. Such request can be made in writing to Juniper Networks, Inc., 1194 N. Mathilda Ave., Sunnyvale, CA 94089, ATTN: General Counsel. You may obtain a copy of the GPL at <http://www.gnu.org/licenses/gpl.html>, and a copy of the LGPL at <http://www.gnu.org/licenses/lgpl.html>.

15. **Miscellaneous.** This Agreement shall be governed by the laws of the State of California without reference to its conflicts of laws principles. The provisions of the U.N. Convention for the International Sale of Goods shall not apply to this Agreement. For any disputes arising under this Agreement, the Parties hereby consent to the personal and exclusive jurisdiction of, and venue in, the state and federal courts within Santa Clara County, California. This Agreement constitutes the entire and sole agreement between Juniper and the Customer with respect to the Software, and supersedes all prior and contemporaneous

agreements relating to the Software, whether oral or written (including any inconsistent terms contained in a purchase order), except that the terms of a separate written agreement executed by an authorized Juniper representative and Customer shall govern to the extent such terms are inconsistent or conflict with terms contained herein. No modification to this Agreement nor any waiver of any rights hereunder shall be effective unless expressly assented to in writing by the party to be charged. If any portion of this Agreement is held invalid, the Parties agree that such invalidity shall not affect the validity of the remainder of this Agreement. This Agreement and associated documentation has been written in the English language, and the Parties agree that the English version will govern. (For Canada: Les parties aux présentes confirment leur volonté que cette convention de même que tous les documents y compris tout avis qui s'y rattache, soient rédigés en langue anglaise. (Translation: The parties confirm that this Agreement and all related documentation is and will be in the English language)).

Abbreviated Table of Contents

	About This Guide	xxv
Part 1	Policy Framework	
Chapter 1	Introduction to Policy Framework	3
Part 2	Routing Policies	
Chapter 2	Introduction to Routing Policy	15
Chapter 3	Routing Policy Configuration Statements	35
Chapter 4	Routing Policy Configuration	39
Chapter 5	Extended Match Conditions Configuration	97
Chapter 6	Extended Actions Configuration	137
Chapter 7	Summary of Routing Policy Configuration Statements	155
Part 3	Firewall Filters	
Chapter 8	Introduction to Firewall Filters	173
Chapter 9	Firewall Filter Configuration	177
Chapter 10	Policer Overview	255
Chapter 11	Policer Configuration	257
Chapter 12	Summary of Firewall Filter and Policer Configuration Statements	287
Part 4	Traffic Sampling, Forwarding and Monitoring	
Chapter 13	Traffic Sampling, Forwarding, and Monitoring Overview	309
Chapter 14	Introduction to Traffic Sampling Configuration	311
Chapter 15	Traffic Forwarding and Monitoring Configuration	325
Chapter 16	Summary of Traffic Sampling, Forwarding, and Monitoring Configuration Statements	349
Part 5	Indexes	
	Index	431
	Index of Statements and Commands	439

Table of Contents

About This Guide	xxv
JUNOS Documentation and Release Notes	xxv
Objectives	xxvi
Audience	xxvi
Supported Platforms	xxvi
Using the Indexes	xxvii
Using the Examples in This Manual	xxvii
Merging a Full Example	xxvii
Merging a Snippet	xxviii
Documentation Conventions	xxviii
Documentation Feedback	xxx
Requesting Technical Support	xxx
Self-Help Online Tools and Resources	xxxi
Opening a Case with JTAC	xxxi

Part 1

Policy Framework

Chapter 1

Introduction to Policy Framework	3
Policy Framework Overview	3
Router Flows Affected by Policies	4
Policy Architecture	6
Control Points	7
Policy Components	7
Default Policies and Actions	8
Configuration Tasks	8
Policy Configuration Recommendations	9
Comparison of Routing Policies and Firewall Filters	9

Part 2**Routing Policies****Chapter 2****Introduction to Routing Policy****15**

Routing Policy Overview	16
Importing and Exporting Routes	17
Protocols That Can Be Imported To and Exported from the Routing Table	18
Routing Tables Affected by Routing Policies	19
Default Routing Policies and Actions	20
Default Import and Export Policies for Protocols	21
Creating Routing Policies	22
Configuring a Routing Policy	23
Routing Policy Match Conditions	24
Routing Policy Named Match Conditions	25
Routing Policy Actions	25
Routing Policy Terms	26
Applying Routing Policy	26
Routing Protocol Support for Import and Export Policy	27
Protocol Support for Import and Export Policies	28
Applying Routing Policy to Routing Protocols	28
Applying Export Policies to the Forwarding Table	29
Evaluating a Routing Policy	29
How a Routing Policy Is Evaluated	29
How a Routing Policy Chain Is Evaluated	30
How a Routing Policy Expression Is Evaluated	31
How a Routing Policy Subroutine Is Evaluated	31
Routing Policy Tests	33

Chapter 3**Routing Policy Configuration Statements****35**

Configuring Routing Policy	35
Minimum Routing Policy Configuration	36
Minimum Routing Policy Chain Configuration	36
Minimum Subroutine Configuration	37

Chapter 4**Routing Policy Configuration****39**

Defining Routing Policies	40
Configuring Match Conditions in Routing Policy Terms	41
Configuring Actions in Routing Policy Terms	47
Configuring Flow Control Actions	48
Configuring Actions That Manipulate Route Characteristics	49
Configuring the Default Action in Routing Policies	54
Example: Configuring the Default Action in a Routing Policy	55
Configuring a Final Action in Routing Policies	56
Logging Matches to a Routing Policy Term	56
Configuring Separate Actions for Routes in Route Lists	57

Applying Routing Policies and Policy Chains to Routing Protocols	57
Effect of Omitting Ingress Match Conditions from Export Policies	58
Applying Policy Expressions to Routes Exported from Routing Tables	59
Policy Expression Examples	61
How a Policy Expression Is Evaluated	62
Example: Evaluating Policy Expressions	63
Applying Routing Policies to the Forwarding Table	64
Configuring Dynamic Routing Policies	65
Configuring Routing Policies and Policy Objects in the Dynamic Database	66
Configuring Routing Policies Based on Dynamic Database Configuration	67
Applying Dynamic Routing Policies to BGP	68
Preventing Reestablishment of BGP Peering Sessions After NSR Routing Engine Switchover	68
Example: Configuring a BGP Export Policy That References a Dynamic Routing Policy	69
Forwarding Packets to the Discard Interface	71
Testing Routing Policies	72
Example: Testing a Routing Policy	72
Routing Policy Examples	72
Example: Defining a Routing Policy from BGP to IS-IS	73
Example: Using Routing Policy to Set a Preference	74
Example: Importing and Exporting Access and Access-Internal Routes in a Routing Policy	74
Example: Exporting Routes to IS-IS	75
Example: Applying Export and Import Policies to BGP Peer Groups	75
Example: Applying a Prefix to Routes Learned from a Peer	76
Example: Redistributing BGP Routes with a Specific Community Tag into IS-IS	76
Example: Redistributing OSPF Routes into BGP	76
Example: Exporting Direct Routes Into IS-IS	77
Example: Exporting Internal IS-IS Level 1 Routes to Level 2	77
Example: Exporting IS-IS Level 2 Routes to Level 1	78
Example: Assigning Different Forwarding Next-Hop LSPs to Different Destination Prefixes	78
Example: Grouping Destination Prefixes	79
Example: Grouping Source Prefixes	80
Example: Grouping Source and Destination Prefixes in a Forwarding Class	81
Example: Accepting Routes with Specific Destination Prefixes	82
Example: Accepting Routes from BGP with a Specific Destination Prefix	83
Example: Using Routing Policy in an ISP Network	83
Requesting a Single Default Route on the Customer 1 Router	85
Requesting Specific Routes on the Customer 2 Router	86
Configuring a Peer Policy on ISP Router 3	88
Configuring Private and Exchange Peers on ISP Router 1 and 2	90
Configuring Locally Defined Static Routes on the Exchange Peer 2 Router	93
Configuring Outbound and Generated Routes on the Private Peer 2 Router	93

Chapter 5**Extended Match Conditions Configuration****97**

Configuring AS Path Regular Expressions to Use as Routing Policy Match Conditions	97
Configuring AS Path Regular Expressions	98
Configuring a Null AS Path	102
How AS Path Regular Expressions Are Evaluated	103
Examples: Configuring AS Path Regular Expressions	103
Overview of BGP Communities and Extended Communities as Routing Policy Match Conditions	104
Defining BGP Communities and Extended Communities for Use in Routing Policy Match Conditions	106
Defining BGP Communities for Use in Routing Policy Match Conditions	106
Using UNIX Regular Expressions in Community Names	107
Defining BGP Extended Communities for Use in Routing Policy Match Conditions	109
Examples: Defining BGP Extended Communities	111
Inverting Community Matches	111
Including BGP Communities and Extended Communities in Routing Policy Match Conditions	111
How BGP Communities and Extended Communities Are Evaluated in Routing Policy Match Conditions	112
Using Routing Policies to Prevent Advertisement of BGP Communities to Neighbors	113
Examples: Configuring BGP Communities as Routing Policy Match Conditions	113
Configuring Prefix Lists for Use in Routing Policy Match Conditions	117
Configuring Prefix Lists	118
How Prefix Lists Are Evaluated in Routing Policy Match Conditions	119
Configuring Prefix List Filters	120
Example: Configuring a Prefix List	120
Configuring Route Lists for Use in Routing Policy Match Conditions	121
Configuring Route Lists	122
How Route Lists Are Evaluated in Routing Policy Match Conditions	124
How Prefix Order Affects Route List Evaluation	125
Common Configuration Problem with the Longest-Match Lookup	125
Route List Examples	126
Example: Rejecting Routes with Specific Destination Prefixes and Mask Lengths	126
Example: Rejecting Routes with a Mask Length Greater than Eight	127
Example: Rejecting Routes with Mask Length Between 26 and 29	127
Example: Rejecting Routes from Specific Hosts	127
Example: Accepting Routes with a Defined Set of Prefixes	128
Example: Rejecting Routes with a Defined Set of Prefixes	128
Example: Rejecting Routes with Prefixes Longer than 24 Bits	129

Example: Rejecting PIM Multicast Traffic Joins	129
Example: Rejecting PIM Traffic	130
Configuring Subroutines in Routing Policy Match Conditions	130
Configuring Subroutines	130
Possible Consequences of Termination Actions in Subroutines	131
Example: Configuring a Subroutine	134
Configuring Routing Policy Match Conditions Based on Routing Table	
Entries	134

Chapter 6

Extended Actions Configuration **137**

Prepending AS Numbers to BGP AS Paths	137
Adding AS Numbers to BGP AS Paths	138
Using Routing Policies to Damp BGP Route Flapping	138
Configuring BGP Flap Damping Parameters	139
Specifying BGP Flap Damping as the Action in Routing Policy Terms	141
Disabling Damping for Specific Address Prefixes	142
Example: Disabling Damping for a Specific Address Prefix	142
Example: Configuring BGP Flap Damping	142
Overview of Per-Packet Load Balancing	144
Configuring Per-Packet Load Balancing	145
Per-Packet Load Balancing Examples	147
Configuring Load Balancing Based on MPLS Labels	147
Configuring Load Balancing for Ethernet Pseudowires	150
Configuring Load Balancing Based on MAC Addresses	151
Configuring VPLS Load Balancing Based on IP and MPLS Information	151
Configuring VPLS Load Balancing on MX Series Ethernet Services	
Routers	153

Chapter 7

Summary of Routing Policy Configuration Statements **155**

apply-path	155
as-path	156
as-path-group	157
community	158
condition	160
damping	161
dynamic-db	162
export	163
import	165
policy-options	166
policy-statement	167
prefix-list	169
prefix-list-filter	170

Part 3**Firewall Filters****Chapter 8****Introduction to Firewall Filters****173**

Firewall Filter Overview	173
Firewall Filter Components	174
Firewall Filter Types	175
Supported Standards	176

Chapter 9**Firewall Filter Configuration****177**

Configuring Firewall Filters	178
Configuring Standard Firewall Filters	179
How Firewall Filters Are Evaluated	182
Overview of Match Conditions in Firewall Filter Terms	183
Configuring IPv4 Match Conditions	183
Configuring IPv6 Match Conditions	187
Configuring Protocol-Independent Match Conditions	191
Configuring Layer 2 Circuit Cross-Connect Match Conditions	191
Configuring MPLS Match Conditions	192
Configuring VPLS Match Conditions	193
Configuring Layer 2 Bridging Match Conditions for MX Series Ethernet Services Routers	197
Overview of Protocol Match Conditions	200
Example: Matching on Destination Port and Protocol Fields	200
Overview of Class-Based Match Conditions	201
How to Specify Firewall Filter Match Conditions	202
Numeric and Text Values in Match Conditions	202
Prefixes in Match Conditions	203
Bit-Field Values in Match Conditions	206
Configuring Actions in Firewall Filter Terms	208
Example: Counting and Sampling Accepted Packets	211
Example: Setting the DSCP Bit to Zero	213
Configuring Nested Firewall Filters	213
Example: Configuring Nested Filters	214
Applying Firewall Filters to Interfaces	215
Configuring Interface-Specific Counters	216
Example: Configuring Interface-Specific Counters	216
Defining Interface Groups	217
Example: Defining Interface Groups	218
Overview of Firewall Filter Lists	219
Firewall Filter Examples	223
Example: Blocking Telnet and SSH Access	223
Example: Blocking TFTP Access	224
Example: Accepting DHCP Packets with Specific Addresses	225
Example: Defining a Policier for a Destination Class	225
Example: Counting IP Option Packets	226
Example: Accepting OSPF Packets from Certain Addresses	227
Example: Matching Packets Based on Two Unrelated Criteria	227

Example: Counting Both Accepted and Rejected Packets	228
Example: Blocking TCP Connections to a Certain Port Except from BGP Peers	228
Example: Accepting Packets with Specific IPv6 TCP Flags	229
Example: Setting a Rate Limit for Incoming Layer 2 Control Packets	230
Configuring Service Filters	231
Configuring Simple Filters	232
Example: Configuring a Simple Filter	233
Configuring Firewall Filters for Logical Systems	234
Guidelines for Firewall Configuration in Logical Systems	235
Scenario 1: Firewall Objects Reference Other Firewall Objects	235
Scenario 2: Nonfirewall Objects Reference Firewall Objects	236
Scenario 3: Firewall Objects Reference Nonfirewall Objects	240
Unsupported Configuration Statements, Actions, and Action Modifiers	242
Configuring Accounting for Firewall Filters	247
Configuring Filter-Based Forwarding	248
Examples: Configuring Filter-Based Forwarding	249
Configuring Forwarding Table Filters	250
Overview of Forwarding Table Filters	250
Configuring a Forwarding Table Filter	251
Configuring System Logging of Firewall Filter Operations	252
Example: Configuring Firewall Filter System Logging	253

Chapter 10**Policer Overview****255****Chapter 11****Policer Configuration****257**

Configuring Policers	257
Minimum Policer Configuration	259
Configuring Policers	259
Configuring Rate Limiting	260
Configuring Policer Actions	261
Example: Configuring a Policer Action	261
Configuring Multifield Classifiers for Policing	262
Configuring Filter-Specific Policers	263
Configuring Policer Actions for Specific Address Prefixes	264
Examples: Configuring Policer Actions for Specific Address Prefixes	265
Examples: Classifying Traffic	268
Configuring Interface Sets	269
Applying Interface Policers	270
Example: Applying an Interface Policer	270

Configuring Aggregate Policers	271
Example: Configuring an Aggregate Policer	271
Configuring a Hierarchical Policar	272
Physical Interface Policar Overview	273
Configuring Physical Interface Policers	273
Configuring Physical Interface Policers	274
Configuring Firewall Filters That Reference Physical Interface Policers	275
Applying Firewall Filters That Reference Physical Interface Policers	276
Configuring Bandwidth Policers	277
Example: Configuring a Bandwidth Policar	277
Configuring Load-Balance Groups	278
Configuring Tricolor Marking	278
Configuring Tricolor Marking Policers	278
Example: Configuring a Tricolor Marking Policar	280
Configuring Interface Policers Using Tricolor Marking Policing	280
Example: Rate-Limiting Bandwidth Using Tricolor Marking Policing	281
Examples: Configuring Policing	282

Chapter 12

Summary of Firewall Filter and Policar Configuration Statements 287

accounting-profile	287
action	288
family	289
filter	290
filter-specific	291
firewall	291
if-exceeding	292
interface-set	293
interface-specific	293
load-balance-group	294
logical-bandwidth-policar	294
logical-interface-policar	295
physical-interface-filter	295
physical-interface-policar	296
policar	297
prefix-action	298
service-filter	299
simple-filter	300
term	301
three-color-policar	303
three-color-policar (Applying)	303
three-color-policar (Configuring)	304
virtual-channel	305

Part 4	Traffic Sampling, Forwarding and Monitoring	
Chapter 13	Traffic Sampling, Forwarding, and Monitoring Overview	309
Chapter 14	Introduction to Traffic Sampling Configuration	311
	Traffic Sampling Configuration	311
	Minimum Traffic Sampling Configuration	312
	Configuring Traffic Sampling	313
	Disabling Traffic Sampling	315
	Configuring the Output File for Traffic Sampling	315
	Traffic Sampling Output Format	316
	Tracing Traffic Sampling Operations	317
	Configuring Flow Aggregation (cflowd)	317
	Debugging cflowd Flow Aggregation	319
	Configuring Active Flow Monitoring Using Version 9	320
	Example: Configuring Active Flow Monitoring Using Version 9	321
	Traffic Sampling Examples	321
	Example: Sampling a Single SONET/SDH Interface	321
	Example: Sampling All Traffic from a Single IP Address	322
	Example: Sampling All FTP Traffic	323
Chapter 15	Traffic Forwarding and Monitoring Configuration	325
	Configuring Traffic Forwarding and Monitoring	325
	Applying Filters to Forwarding Tables	329
	Configuring IPv6 Accounting	330
	Configuring Discard Accounting	330
	Configuring Flow Monitoring	332
	Configuring Next-Hop Groups	333
	Per-Flow and Per-Prefix Load Balancing Overview	333
	Configuring Per-Prefix Load Balancing	334
	Configuring Per-Flow Load Balancing Based on Hash Values	335
	Configuring Routers, Switches, and Interfaces as DHCP and BOOTP Relay Agents	335
	Configuring DNS and TFTP Packet Forwarding	337
	Tracing BOOTP, DNS, and TFTP Forwarding Operations	338
	Configuring the Log Filename	339
	Configuring the Number and Size of Log Files	339
	Configuring Access to the Log File	339
	Configuring a Regular Expression for Lines to Be Logged	340
	Example: Configuring DNS Packet Forwarding	340
	Preventing DHCP Spoofing on MX Series Ethernet Services Routers	340

Configuring Port Mirroring	341
Configuration Guidelines	342
Configuring Port Mirroring	343
Configuring the Port-Mirroring Address Family and Interface	343
Configuring Multiple Port-Mirroring Instances	343
Configuring Port-Mirroring Instances	344
Associating a Port-Mirroring Instance on M320 Routers	344
Associating a Port-Mirroring Instance on M120 Routers	345
Configuring MX Series Ethernet Services Routers and M120 Routers to Mirror Traffic Only Once	345
Configuring Packet Capture	345

Chapter 16

Summary of Traffic Sampling, Forwarding, and Monitoring Configuration Statements 349

accounting	350
aggregation	351
autonomous-system-type	352
bootp	353
cflowd	354
cflowd (Discard Accounting)	355
cflowd (Flow Monitoring)	356
cflowd (Sampling)	357
client-response-ttl	358
description	358
dhcp-relay (DHCP Spoofing Prevention)	359
disable	359
domain	360
export-format	360
family	361
family (Filtering)	362
family (Monitoring)	363
family (Port Mirroring)	364
family (Sampling)	365
family inet	365
family mpls	366
family multiservice	368
file	370
file (Extended DHCP Relay Agent and Helpers Trace Options)	371
file (Packet Capture)	371
file (Sampling)	372
file (Trace Options)	372
filename	373
filename (Packet Capture)	373
filename (Sampling)	373
files	374
files (Packet Capture)	374
files (Sampling and Traceoptions)	374

filter	375
filter (IPv4, IPv6, and MPLS)	375
filter (VPLS)	375
flood	376
flow-active-timeout	376
flow-export-destination	377
flow-inactive-timeout	377
forwarding-options	378
group (DHCP Spoofing Prevention)	378
hash-key	379
helpers	382
indexed-next-hop	384
input	385
input (Forwarding Table)	386
input (Port Mirroring)	386
input (Sampling)	387
instance	388
interface	389
interface (Accounting or Sampling)	389
interface (BOOTP)	390
interface (DHCP Spoofing Prevention)	391
interface (DNS and TFTP Packet Forwarding or Relay Agent)	391
interface (Monitoring)	392
interface (Next-Hop Group)	392
interface (Port Mirroring)	393
load-balance	394
local-dump	395
max-packets-per-second	395
maximum-capture-size	396
maximum-hop-count	396
maximum-packet-length	397
minimum-wait-time	397
mirror-once	398
monitoring	399
next-hop	400
next-hop-group	401
no-filter-check	402
no-listen	402
no-local-dump	402
no-stamp	403
no-world-readable	403
output	404
output (Accounting)	405
output (Forwarding Table)	406
output (Monitoring)	406
output (Port Mirroring)	407
output (Sampling)	408
packet-capture	410
per-flow	410
per-prefix	411
port	412

port-mirroring	413
rate	414
route-accounting	415
run-length	415
sampling	416
server	418
server (DHCP and BOOTP Relay Agent)	419
server (DNS and TFTP Service)	419
size	420
size (Packet Capture)	421
size (Sampling and Traceoptions)	421
stamp	422
tftp	422
traceoptions	423
traceoptions (DNS and TFTP Packet Forwarding)	424
traceoptions (Port Mirroring and Traffic Sampling)	426
version	426
version9	427
world-readable	427

Part 5

Indexes

Index	431
Index of Statements and Commands	439

List of Figures

Part 1

Policy Framework

Chapter 1	Introduction to Policy Framework	3
	Figure 1: Flows of Routing Information and Packets	5
	Figure 2: Routing Policies to Control Routing Information Flow	5
	Figure 3: Firewall Filters to Control Packet Flow	6
	Figure 4: Policy Control Points	7

Part 2

Routing Policies

Chapter 2	Introduction to Routing Policy	15
	Figure 5: Importing and Exporting Routes	17
	Figure 6: Importing and Exporting Routing Policies	23
	Figure 7: Routing Policy Components	23
	Figure 8: Routing Policy Evaluation	30
	Figure 9: Routing Policy Chain Evaluation	31
	Figure 10: Routing Policy Subroutine Evaluation	33
Chapter 4	Routing Policy Configuration	39
	Figure 11: ISP Network Example	84

Part 3

Firewall Filters

Chapter 11	Policer Configuration	257
	Figure 12: Incoming and Outgoing Interface Policers	270

Part 4

Traffic Sampling, Forwarding and Monitoring

Chapter 14	Introduction to Traffic Sampling Configuration	311
	Figure 13: Configure Sampling Rate	314

List of Tables

About This Guide	xxv
Table 1: Notice Icons	xxix
Table 2: Text and Syntax Conventions	xxix

Part 1

Policy Framework

Chapter 1	Introduction to Policy Framework	3
	Table 3: Purpose of Routing Policies and Firewall Filters	9
	Table 4: Implementation Differences Between Routing Policies and Firewall Filters	10

Part 2

Routing Policies

Chapter 2	Introduction to Routing Policy	15
	Table 5: Protocols That Can Be Imported To and exported from the Routing Table	18
	Table 6: Routing Tables Affected by Routing Policies	19
	Table 7: Default Import and Export Policies for Protocols	21
	Table 8: Match Conditions	24
	Table 9: Protocol Support for Import and Export Policies	28
Chapter 4	Routing Policy Configuration	39
	Table 10: Routing Policy Match Conditions	42
	Table 11: Flow Control Actions	49
	Table 12: Actions That Manipulate Route Characteristics	49
	Table 13: Policy Action Conversion Values	60
	Table 14: Policy Expression Logical Operators	60
Chapter 5	Extended Match Conditions Configuration	97
	Table 15: AS Path Regular Expression Operators	100
	Table 16: Examples of AS Path Regular Expressions	100
	Table 17: Community Attribute Regular Expression Operators	108
	Table 18: Examples of Community Attribute Regular Expressions	109
	Table 19: Prefix List and Route List Differences	118
	Table 20: Route List Match Types for a Prefix List Filter	120
	Table 21: Route List Match Types for a Prefix List	123
	Table 22: Match Type Examples	123
Chapter 6	Extended Actions Configuration	137
	Table 23: Damping Parameters	140

Part 3**Firewall Filters**

Chapter 9	Firewall Filter Configuration	177
	Table 24: IPv4 Firewall Filter Match Conditions	183
	Table 25: IPv6 Firewall Filter Match Conditions	188
	Table 26: Protocol-Independent Firewall Filter Match Conditions	191
	Table 27: Layer 2 Circuit Cross-Connect Firewall Filter Match Conditions	191
	Table 28: MPLS Firewall Filter Match Conditions	192
	Table 29: VPLS Firewall Filter Match Conditions	193
	Table 30: Layer 2 Bridging Firewall Filter Match Conditions (MX Series Ethernet Services Routers Only)	197
	Table 31: Bit-Field Logical Operators	207
	Table 32: Firewall Filter Actions	209
	Table 33: Unsupported Firewall Statements for Logical Systems	242
	Table 34: Unsupported Firewall Actions and Action Modifiers for Logical Systems	243

About This Guide

This preface provides the following guidelines for using the *JUNOS® Software Policy Framework Configuration Guide*:

- JUNOS Documentation and Release Notes on page xxv
- Objectives on page xxvi
- Audience on page xxvi
- Supported Platforms on page xxvi
- Using the Indexes on page xxvii
- Using the Examples in This Manual on page xxvii
- Documentation Conventions on page xxviii
- Documentation Feedback on page xxx
- Requesting Technical Support on page xxx

JUNOS Documentation and Release Notes

For a list of related JUNOS documentation, see <http://www.juniper.net/techpubs/software/junos/>.

If the information in the latest release notes differs from the information in the documentation, follow the *JUNOS Software 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/>.

Juniper Networks supports a technical book program to publish books by Juniper Networks engineers and subject matter experts with book publishers around the world. These books go beyond the technical documentation to explore the nuances of network architecture, deployment, and administration using JUNOS Software and Juniper Networks devices. In addition, the Juniper Networks Technical Library, published in conjunction with O'Reilly Media, explores improving network security, reliability, and availability using JUNOS configuration techniques. All the books are for sale at technical bookstores and book outlets around the world. The current list can be viewed at <http://www.juniper.net/books>.

Objectives

This guide is designed for network administrators who are configuring and monitoring a Juniper Networks J Series Services Routers, M Series Multiservice Edge Routers, MX Series Ethernet Services Routers, or T Series Core Routers.



NOTE: For additional information about JUNOS Software—either corrections to or information that might have been omitted from this guide—see the software release notes at <http://www.juniper.net/>.

Audience

This guide is designed for network administrators who are configuring and monitoring a Juniper Networks M Series, MX Series, T Series, EX Series, or J Series router or switch.

To use this guide, you need a broad understanding of networks in general, the Internet in particular, networking principles, and network configuration. You must also be familiar with one or more of the following Internet routing protocols:

- Border Gateway Protocol (BGP)
- Distance Vector Multicast Routing Protocol (DVMRP)
- Intermediate System-to-Intermediate System (IS-IS)
- Internet Control Message Protocol (ICMP) router discovery
- Internet Group Management Protocol (IGMP)
- Multiprotocol Label Switching (MPLS)
- Open Shortest Path First (OSPF)
- Protocol-Independent Multicast (PIM)
- Resource Reservation Protocol (RSVP)
- Routing Information Protocol (RIP)
- Simple Network Management Protocol (SNMP)

Personnel operating the equipment must be trained and competent; must not conduct themselves in a careless, willfully negligent, or hostile manner; and must abide by the instructions provided by the documentation.

Supported Platforms

For the features described in this manual, JUNOS Software currently supports the following platforms:

- J Series
- M Series

- MX Series
- T Series
- EX Series

Using the Indexes

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

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

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

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. 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.xml; }

```

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

```

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

```

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

```

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

```

For more information about the `load` command, see the *JUNOS CLI User Guide*.

Documentation Conventions

Table 1 on page xxix 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 xxix 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 <code>configure</code> 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
<i>Italic text like this</i>	<ul style="list-style-type: none"> Introduces important new terms. Identifies book names. Identifies RFC and Internet draft titles. 	<ul style="list-style-type: none"> A policy <i>term</i> is a named structure that defines match conditions and actions. <i>JUNOS System Basics Configuration Guide</i> RFC 1997, <i>BGP Communities Attribute</i>
<i>Italic text like this</i>	Represents variables (options for which you substitute a value) in commands or configuration statements.	Configure the machine's domain name: [edit] root@# set system domain-name <i>domain-name</i>
Plain text like this	Represents names of configuration statements, commands, files, and directories; IP addresses; configuration hierarchy levels; or labels on routing platform components.	<ul style="list-style-type: none"> To configure a stub area, include the stub statement at the [edit protocols ospf area area-id] hierarchy level. The console port is labeled CONSOLE.
< > (angle brackets)	Enclose optional keywords or variables.	stub <default-metric <i>metric</i> >;

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

Convention	Description	Examples
(pipe symbol)	Indicates a choice between the mutually exclusive keywords or variables on either side of the symbol. The set of choices is often enclosed in parentheses for clarity.	broadcast multicast (string1 string2 string3)
# (pound sign)	Indicates a comment specified on the same line as the configuration statement to which it applies.	rsvp { # Required for dynamic MPLS only
[] (square brackets)	Enclose a variable for which you can substitute one or more values.	community name members [community-ids]
Indentation 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.	<ul style="list-style-type: none">■ In the Logical Interfaces box, select All Interfaces.■ To cancel the configuration, click Cancel.
> (bold right angle bracket)	Separates levels in a hierarchy of J-Web selections.	In the configuration editor hierarchy, select Protocols > Ospf .

Documentation Feedback

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

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

Requesting Technical Support

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

contract, or are covered under warranty, and need postsales 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/customers/support/downloads/710059.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, visit us at <http://www.juniper.net/support/requesting-support.html>

Part 1

Policy Framework

- Introduction to Policy Framework on page 3

Chapter 1

Introduction to Policy Framework

This chapter discusses the following topics related to understanding the JUNOS policy framework:

- Policy Framework Overview on page 3
- Router Flows Affected by Policies on page 4
- Policy Architecture on page 6
- Control Points on page 7
- Policy Components on page 7
- Default Policies and Actions on page 8
- Configuration Tasks on page 8
- Policy Configuration Recommendations on page 9
- Comparison of Routing Policies and Firewall Filters on page 9

Policy Framework Overview

The JUNOS Software provides a *policy framework*, which is a collection of JUNOS 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

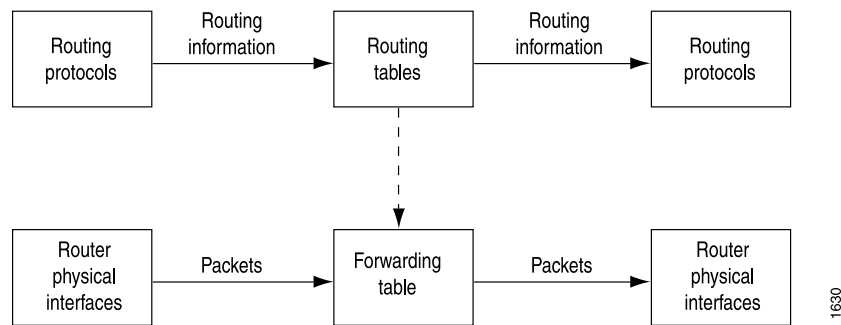
The JUNOS policies affect the following router flows:

- 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.

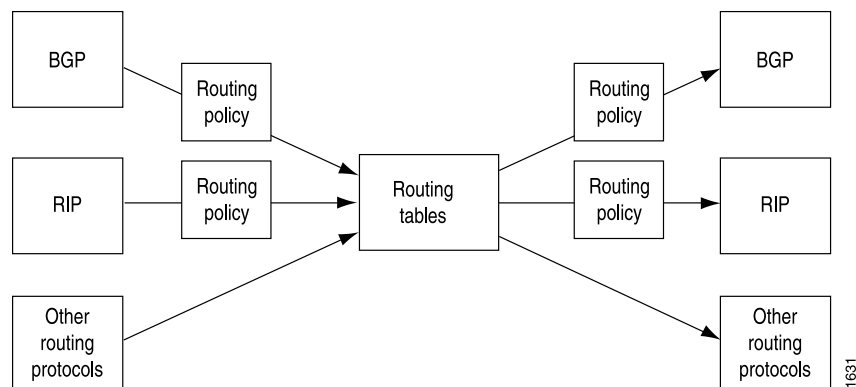
Figure 1 on page 5 illustrates the flows through the router. Although the flows are very different from each other, they are also interdependent. Routing policies determine which routes are placed in the forwarding table. The forwarding table, in turn, has an integral role in determining the appropriate 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. For information about the active route selection process, see the *JUNOS Routing Protocols Configuration Guide*.)

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

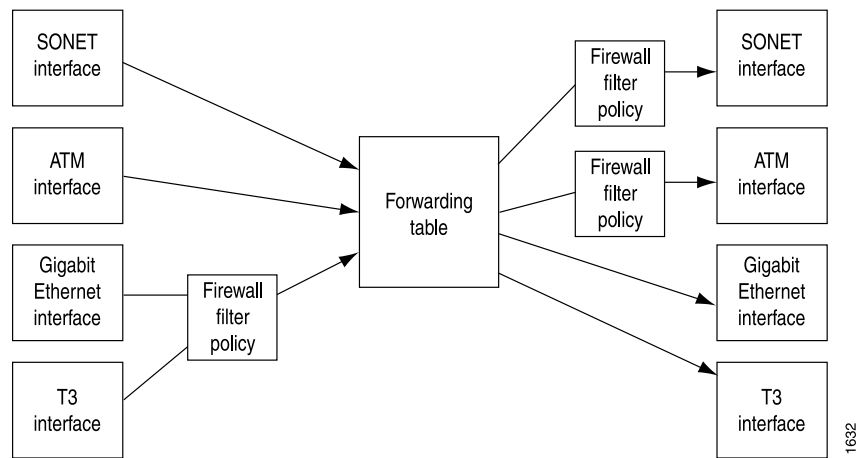
You can configure firewall filters to control the following (see Figure 3 on page 6):

- 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 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 policy framework share the following architecture and configuration fundamentals:

- Control Points on page 7
- Policy Components on page 7
- Default Policies and Actions on page 8
- Configuration Tasks on page 8
- Policy Configuration Recommendations on page 9



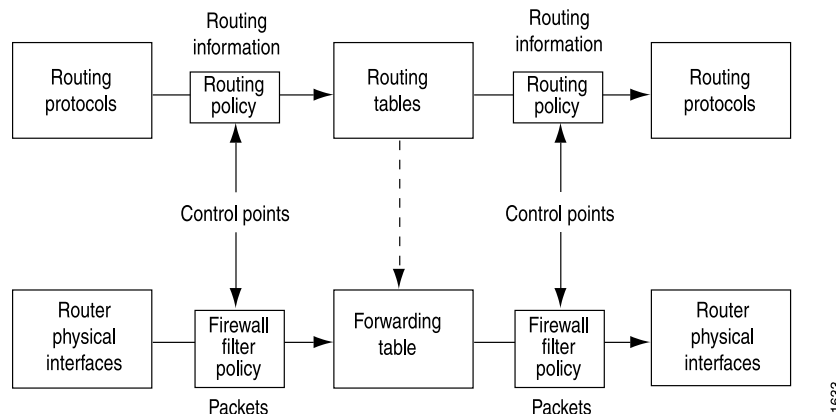
NOTE: This section highlights the fundamental architecture that all policies share. Note, however, that the implementation details of routing policies and firewall filters are very different. For information about these differences, see “Comparison of Routing Policies and Firewall Filters” on page 9.

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 23 and “Firewall Filter Components” on page 174.

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 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 10 compares the implementation details for routing policies and firewall filters, highlighting the similarities and differences in their configuration.

For complete information about routing policies, see “Routing Policies” on page 13. For complete information about firewall filters, see “Firewall Filters” on page 171.

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 a policy that contains terms, match conditions, and actions.	Define a policy that contains terms, match conditions, and actions.
<ul style="list-style-type: none"> ■ Define policy ■ Apply policy 	<p>Apply one or more export or import policies to a routing protocol. You can also apply a <i>policy expression</i>, which uses Boolean logical operators with multiple import or export policies.</p> <p>You can also apply one or more export policies to the forwarding table.</p>	<p>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).</p> <p>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.</p>
Terms	<p>Configure as many terms as desired. Define a name for each term.</p> <p>Terms are evaluated in the order in which you specify them.</p> <p>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.</p>	<p>Configure as many terms as desired. Define a name for each term.</p> <p>Terms are evaluated in the order in which you specify them.</p> <p>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.</p>
Match conditions	<p>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:</p> <ul style="list-style-type: none"> ■ 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. 	<p>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:</p> <ul style="list-style-type: none"> ■ 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	<p>Specify zero or one action to take if a route matches all criteria. You can specify the following actions:</p> <ul style="list-style-type: none"> ■ 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. <p>In addition to the preceding actions, you can also specify zero or more of the following types of actions:</p> <ul style="list-style-type: none"> ■ 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. 	<p>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:</p> <ul style="list-style-type: none"> ■ 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. <p>In addition to zero or the preceding actions, you can also specify zero or more action modifiers. You can specify the following action modifiers:</p> <ul style="list-style-type: none"> ■ 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	<p>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.</p> <p>The following default actions exist for routing policies:</p> <ul style="list-style-type: none"> ■ 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: <ul style="list-style-type: none"> ■ 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. 	<p>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).</p> <p>The following default actions exist for firewall filters:</p> <ul style="list-style-type: none"> ■ 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.

Part 2

Routing Policies

- Introduction to Routing Policy on page 15
- Routing Policy Configuration Statements on page 35
- Routing Policy Configuration on page 39
- Extended Match Conditions Configuration on page 97
- Extended Actions Configuration on page 137
- Summary of Routing Policy Configuration Statements on page 155

Chapter 2

Introduction to Routing Policy

This chapter discusses the following topics related to understanding and creating routing policies:

- Routing Policy Overview on page 16
- Importing and Exporting Routes on page 17
- Protocols That Can Be Imported To and Exported from the Routing Table on page 18
- Routing Tables Affected by Routing Policies on page 19
- Default Routing Policies and Actions on page 20
- Default Import and Export Policies for Protocols on page 21
- Creating Routing Policies on page 22
- Configuring a Routing Policy on page 23
- Routing Policy Match Conditions on page 24
- Routing Policy Named Match Conditions on page 25
- Routing Policy Actions on page 25
- Routing Policy Terms on page 26
- Applying Routing Policy on page 26
- Routing Protocol Support for Import and Export Policy on page 27
- Protocol Support for Import and Export Policies on page 28
- Applying Routing Policy to Routing Protocols on page 28
- Applying Export Policies to the Forwarding Table on page 29
- Evaluating a Routing Policy on page 29
- How a Routing Policy Is Evaluated on page 29
- How a Routing Policy Chain Is Evaluated on page 30
- How a Routing Policy Expression Is Evaluated on page 31
- How a Routing Policy Subroutine Is Evaluated on page 31
- Routing Policy Tests on page 33

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 Software maintains, the discussion in the rest of this chapter assumes the `inet.0` routing table unless explicitly stated otherwise. By default, the JUNOS Software 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 19.

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 Software 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. For information about the active route selection process, see the *JUNOS Routing Protocols Configuration Guide*.)
- 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 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 Software 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 65.

Importing and Exporting Routes

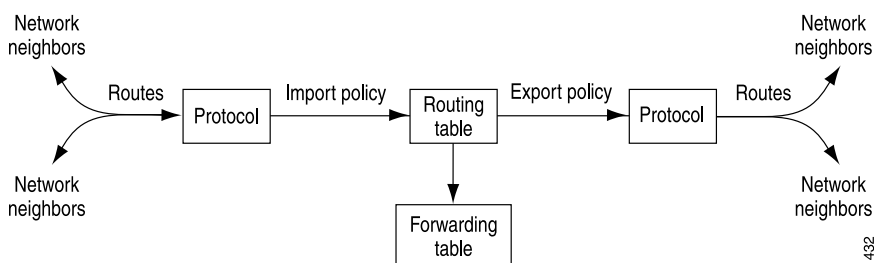
Two terms—*import* and *export*—explain how routes move between the routing protocols and the routing table (see Figure 5 on page 17):

- 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. For more information about the active path selection algorithm, see the *JUNOS Routing Protocols Configuration Guide*.



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. For more information about advertising inactive routes, see the *JUNOS Routing Protocols Configuration Guide*.

Table 5 on page 18 lists the routing protocols from which the routing table can import routes and to which the routing table can export routes. Table 5 on page 18 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 Table 7 on page 21. 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 Table 5 on page 18.

Protocols That Can Be Imported To and Exported from the Routing Table

Table 5: 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

Table 5: Protocols That Can Be Imported To and exported from the Routing Table (continued)

Protocol	Import	Export
OSPF	Yes	Yes
Protocol Independent Multicast (PIM) dense mode	Yes	Yes
PIM sparse mode	Yes	Yes
PIM sparse-dense mode	Yes	Yes
Pseudoprotocol:	Yes	No
<ul style="list-style-type: none"> ■ Direct routes ■ Explicitly configured routes <ul style="list-style-type: none"> ■ Aggregate routes ■ Generated routes ■ Local routes ■ Static routes 		
Routing Information Protocol (RIP) and Routing Information Protocol next generation (RIPng)	Yes	Yes

Routing Tables Affected by Routing Policies

Table 6 on page 19 lists the routing tables affected by default and user-defined routing policies and the types of routes that each routing table stores.

Table 6: Routing Tables Affected by Routing Policies

Routing Table	Type of Routes Stored
inet.0	Unicast IPv4 routes
.inet.0 <i>instance-name</i>	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 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.

For more information about routing tables, see the *JUNOS Routing Protocols Configuration Guide*.

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. Table 7 on page 21 summarizes the for each routing protocol that imports and exports routes. The actions in the 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 Software 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 Table 5 on page 18. 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 Table 9 on page 28.

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. Table 7 on page 21 includes information about the routing tables used by each protocol.

Default Import and Export Policies for Protocols

Table 7: Default Import and Export Policies for Protocols

Importing or Exporting Protocol	Default Import Policy	Default Export Policy
BGP	Accept all BGP IPv4 routes learned from configured neighbors and import into the <code>inet.0</code> routing table. Accept all BGP IPv6 routes learned from configured neighbors and import into the <code>inet6.0</code> routing table.	Accept and export active BGP routes.
DVMRP	Accept all DVMRP routes and import into the <code>inet.1</code> routing table.	Accept and export active DVMRP routes.
IS-IS	Accept all IS-IS routes and import into the <code>inet.0</code> and <code>inet6.0</code> 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 <code>inet.3</code> routing table.	Reject everything.

Table 7: Default Import and Export Policies for Protocols *(continued)*

Importing or Exporting Protocol	Default Import Policy	Default Export Policy
MPLS	Accept all MPLS routes and import into the <code>inet.3</code> routing table.	Accept and export active MPLS routes.
OSPF	Accept all OSPF routes and import into the <code>inet.0</code> 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 <code>inet.1</code> routing table.	Accept active PIM dense mode routes.
PIM sparse mode	Accept all PIM sparse mode routes and import into the <code>inet.1</code> 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 <code>inet.0</code> 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.
RIP	Accept all RIP routes learned from configured neighbors and import into the <code>inet.0</code> 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 <code>inet6.0</code> 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 33.	

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:

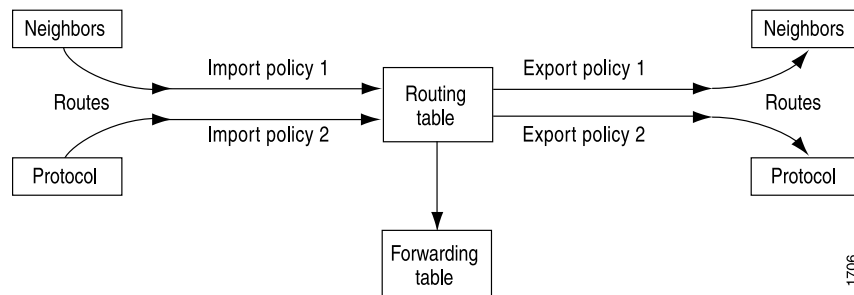
- You do not want a protocol to import all routes into the routing table. If the routing table does not learn about certain routes, they can never be used to forward packets and they can never be redistributed into other routing protocols.
- You do not want a routing protocol to export all the active 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 23, 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

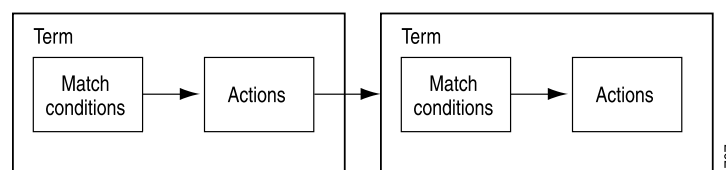


To create a routing policy, you must define the following components:

- **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 23 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 Match Conditions on page 24
- Routing Policy Named Match Conditions on page 25
- Routing Policy Actions on page 25
- Routing Policy Terms on page 26
- Applying Routing Policy 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. For more information about them, see “Extended Match Conditions Configuration” on page 97. 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 8 on page 24 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 Table 10 on page 42.

Table 8: 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.
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.	<p>Actions can be performed on the entire group.</p> <p>You can create multiple communities associated with a particular destination.</p> <p>You can create match conditions using regular expressions.</p>

Table 8: Match Conditions (*continued*)

Match Condition	Category	When to Use	Notes
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	<p>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.</p> <p>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.</p>	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 31.

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 Table 8 on page 24.

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 Table 12 on page 49.

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 Routing Policy

After defining a routing policy, as discussed in “Routing Policy Match Conditions” on page 24 and “Routing Policy Actions” on page 25, 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 Policy on page 27
- Protocol Support for Import and Export Policies on page 28

- Applying Routing Policy to Routing Protocols on page 28
- Applying Export Policies to the Forwarding Table on page 29

Routing Protocol Support for Import and Export Policy

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. Table 9 on page 28 summarizes the import and export policy support for each routing protocol. Table 5 on page 18 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 more information about RIP and RIPng, see the *JUNOS Routing Protocols Configuration Guide*.

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 Table 5 on page 18. For information about the default routing policies for each routing protocol, see Table 7 on page 21.

Protocol Support for Import and Export Policies

Table 9: 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
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

Applying 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 “Overview of Per-Packet Load Balancing” on page 144. For more information about CoS, see the *JUNOS Class of Service Configuration Guide*.

Evaluating a Routing Policy

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

- How a Routing Policy Is Evaluated on page 29
- How a Routing Policy Chain Is Evaluated on page 30
- How a Routing Policy Expression Is Evaluated on page 31
- How a Routing Policy Subroutine Is Evaluated on page 31

For specific information about how the various match conditions are evaluated, see “Configuring Match Conditions in Routing Policy Terms” on page 41 and “Extended Match Conditions Configuration” on page 97.

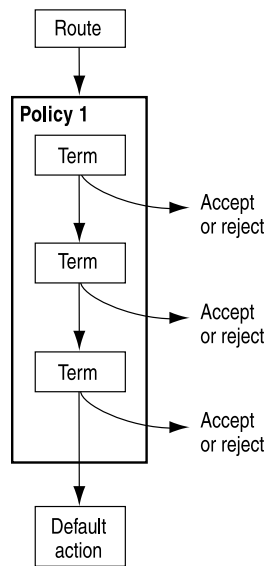
How a Routing Policy Is Evaluated

Figure 8 on page 30 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 20.

Figure 8: Routing Policy Evaluation



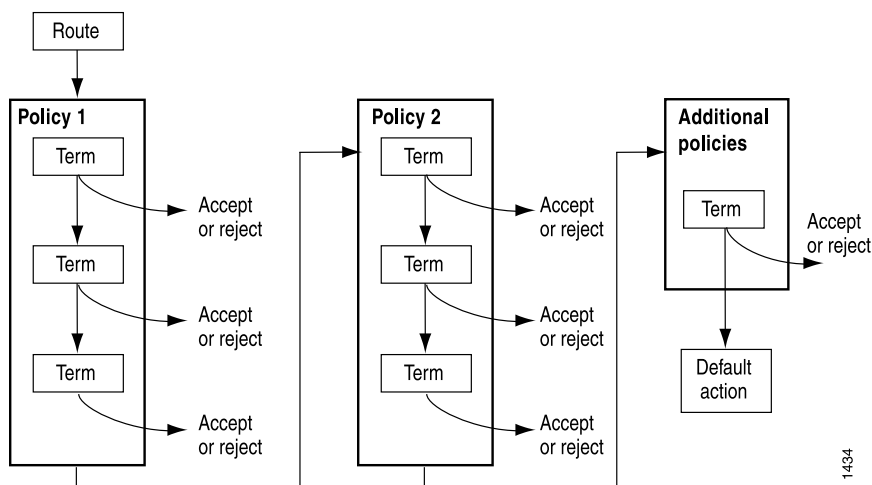
1708

How a Routing Policy Chain Is Evaluated

Figure 9 on page 31 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 20.

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 59.

How a Routing Policy Subroutine Is Evaluated

Figure 10 on page 33 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 one of the actions is meant to manipulate route characteristics, the characteristic is changed regardless of whether `accept`, `reject`, or neither action is specified.

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 “Possible Consequences of Termination Actions in Subroutines” on page 131. For more information about the default routing policies, see “Default Routing Policies and Actions” on page 20.)

- 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 30.

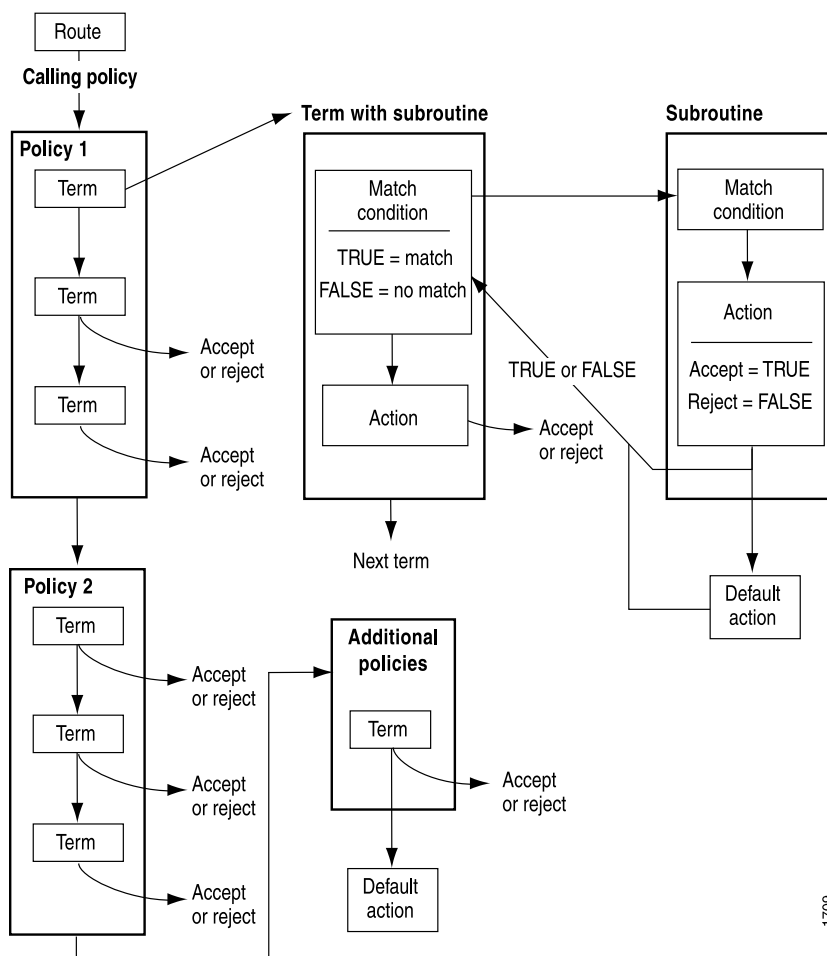


NOTE: If you specify a policy chain as a subroutine, the entire chain acts as a single subroutine. As with other chains, the action specified by the default policy is taken only when the entire chain does not accept or reject a route.



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

Routing Policy Configuration Statements

This section includes the following minimum configurations:

- Configuring Routing Policy on page 35
- Minimum Routing Policy Configuration on page 36
- Minimum Routing Policy Chain Configuration on page 36
- Minimum Subroutine Configuration on page 37

Configuring Routing Policy

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

```
policy-options {  
  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 destination-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 {
        import [ policy-names ];
        export [ 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 destination-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 destination-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 destination-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 destination-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 4

Routing Policy Configuration

This chapter describes the following tasks for configuring routing policies and provides the following examples:

- Defining Routing Policies on page 40
- Configuring Match Conditions in Routing Policy Terms on page 41
- Configuring Actions in Routing Policy Terms on page 47
- Applying Routing Policies and Policy Chains to Routing Protocols on page 57
- Applying Policy Expressions to Routes Exported from Routing Tables on page 59
- Applying Routing Policies to the Forwarding Table on page 64
- Configuring Dynamic Routing Policies on page 65
- Forwarding Packets to the Discard Interface on page 71
- Testing Routing Policies on page 72
- Routing Policy Examples on page 72
- Example: Defining a Routing Policy from BGP to IS-IS on page 73
- Example: Using Routing Policy to Set a Preference on page 74
- Example: Importing and Exporting Access and Access-Internal Routes in a Routing Policy on page 74
- Example: Exporting Routes to IS-IS on page 75
- Example: Applying Export and Import Policies to BGP Peer Groups on page 75
- Example: Applying a Prefix to Routes Learned from a Peer on page 76
- Example: Redistributing BGP Routes with a Specific Community Tag into IS-IS on page 76
- Example: Redistributing OSPF Routes into BGP on page 76
- Example: Exporting Direct Routes Into IS-IS on page 77
- Example: Exporting Internal IS-IS Level 1 Routes to Level 2 on page 77
- Example: Exporting IS-IS Level 2 Routes to Level 1 on page 78
- Example: Assigning Different Forwarding Next-Hop LSPs to Different Destination Prefixes on page 78
- Example: Grouping Destination Prefixes on page 79
- Example: Grouping Source Prefixes on page 80

- Example: Grouping Source and Destination Prefixes in a Forwarding Class on page 81
- Example: Accepting Routes with Specific Destination Prefixes on page 82
- Example: Accepting Routes from BGP with a Specific Destination Prefix on page 83
- Example: Using Routing Policy in an ISP Network on page 83
- Requesting a Single Default Route on the Customer 1 Router on page 85
- Requesting Specific Routes on the Customer 2 Router on page 86
- Configuring a Peer Policy on ISP Router 3 on page 88
- Configuring Private and Exchange Peers on ISP Router 1 and 2 on page 90
- Configuring Locally Defined Static Routes on the Exchange Peer 2 Router on page 93
- Configuring Outbound and Generated Routes on the Private Peer 2 Router on page 93

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 destination-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, recommend that you name all terms.

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

- Configuring Match Conditions in Routing Policy Terms on page 41
- Configuring Actions in Routing Policy Terms on page 47

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 “Effect of Omitting Ingress Match Conditions from Export Policies” on page 58.

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 10 on page 42 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 117 and “Configuring Route Lists for Use in Routing Policy Match Conditions” on page 121. 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 10 on page 42 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 10 on page 42 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. For more information about them, see “Extended Match Conditions Configuration” on page 97. The standard match conditions include criteria that are defined within a routing policy and are less complex than the extended match conditions.

For examples of how to use the **from** and **to** statements, see “Routing Policy Examples” on page 72.

Table 10: 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.	

Table 10: Routing Policy Match Conditions (continued)

Match Condition	Match Condition Category	from Statement Description	to Statement Description
area <i>area-id</i>	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 <i>name</i>	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 97.	
as-path-group <i>group-name</i>	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 97.	
color <i>preference</i> color2 <i>preference</i>	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. For more information about preference values, see the <i>JUNOS Routing Protocols Configuration Guide</i> .	
community [<i>names</i>]	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 104.	
external [<i>type</i> <i>metric-type</i>]	Standard	(OSPF only) Match external routes, including routes exported from one level to another. type is an optional keyword. metric can either be 1 or 2. When you do not specify type , this condition matches all external routes. When you specify type , this condition matches only OSPF routes with the specified OSPF metric type.	
family <i>family-name</i>	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 <i>instance-name</i>	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.
interface <i>interface-name</i>	Standard	Name or IP address of one or more router 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 router 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.
internal	Standard	Match a routing policy against the internal flag for simplified next-hop self policies.	

Table 10: Routing Policy Match Conditions (continued)

Match Condition	Match Condition Category	from Statement Description	to Statement Description
level <i>level</i>	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 <i>value</i>	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 <i>metric metric2 metric3 metric4 metric</i>	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 (<i>scoping-name number</i>) < (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 <i>JUNOS Multicast Protocols Configuration Guide</i> . You can apply this scoping policy to the routing table by including the scope-policy statement at the [edit routing-options] hierarchy level. The <i>number</i> 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: <ul style="list-style-type: none"> ■ 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 10: Routing Policy Match Conditions (continued)

Match Condition	Match Condition Category	from Statement Description	to Statement Description
neighbor <i>address</i>	Standard	<p>Address of one or more neighbors (peers).</p> <p>For BGP, the address can be a directly connected or indirectly connected peer.</p> <p>For all other protocols, the address is the neighbor from which the advertisement is received.</p> <p>NOTE: The <i>neighbor address</i> match condition is not valid for the Routing Information Protocol (RIP).</p>	<p>Address of one or more neighbors (peers).</p> <p>For BGP import policies, specifying <i>to neighbor</i> produces the same result as specifying <i>from neighbor</i>.</p> <p>For BGP export policies, specifying the <i>neighbor</i> match condition has no effect and is ignored.</p> <p>For all other protocols, the <i>to</i> statement matches the neighbor to which the advertisement is sent.</p> <p>NOTE: The <i>neighbor address</i> match condition is not valid for the Routing Information Protocol (RIP).</p>
next-hop <i>address</i>	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.	
next-hop-type merged	Standard	LDP generates next hop based on RSVP and IP next hops available to use combined with the forwarding-class mapping.	You cannot specify this match condition.
origin <i>value</i>	Standard	<p>(BGP only) BGP origin attribute, which is the origin of the AS path information. The value can be one of the following:</p> <ul style="list-style-type: none"> ■ <i>egp</i>—Path information originated in another AS. ■ <i>igp</i>—Path information originated within the local AS. ■ <i>incomplete</i>—Path information was learned by some other means. 	
policy [<i>policy-name</i>]	Extended	<p>Name of a policy to evaluate as a subroutine.</p> <p>For information about this extended match condition, see “Configuring Subroutines in Routing Policy Match Conditions” on page 130.</p>	
preference <i>preference</i> <i>preference2</i> <i>preference</i>	Standard	<p>Preference value. You can specify a primary preference value (<i>preference</i>) and a secondary preference value (<i>preference2</i>). 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.</p> <p>To specify even finer-grained preference values, see the <i>color</i> and <i>color2</i> match conditions in this table.</p> <p>For more information about preference values, see the <i>JUNOS Routing Protocols Configuration Guide</i>.</p>	

Table 10: Routing Policy Match Conditions (*continued*)

Match Condition	Match Condition Category	from Statement Description	to Statement Description
<code>prefix-list</code> <code>prefix-list-name</code> <code>ip-addresses</code>	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 117.	You cannot specify this match condition.
<code>prefix-list-filter</code> <code>prefix-list-name</code> <code>match-type</code>	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 117.	You cannot specify this match condition.
<code>protocol protocol</code>	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: <code>access</code> , <code>access-internal</code> , <code>aggregate</code> , <code>bgp</code> , <code>direct</code> , <code>dvmrp</code> , <code>isis</code> , <code>local</code> , <code>ospf</code> , <code>ospf2</code> , <code>ospf3</code> , <code>pim-dense</code> , <code>pim-sparse</code> , <code>rip</code> , <code>ripng</code> , or <code>static</code> . NOTE: The <code>ospf2</code> statement matches on OSPFv2 routes. The <code>ospf3</code> statement matches on OSPFv3 routes. The <code>ospf</code> 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 74.	
<code>rib routing-table</code>	Standard	Name of a routing table. The value of <i>routing-table</i> can be one of the following: <ul style="list-style-type: none"> ■ <code>inet.0</code>—Unicast IPv4 routes ■ <code>instance-name inet.0</code>—Unicast IPv4 routes for a particular routing instance ■ <code>inet.1</code>—Multicast IPv4 routes ■ <code>inet.2</code>—Unicast IPv4 routes for multicast reverse-path forwarding (RPF) lookup ■ <code>inet.3</code>—MPLS routes ■ <code>mpls.0</code>—MPLS routes for label-switched path (LSP) next hops ■ <code>inet6.0</code>—Unicast IPv6 routes 	
<code>route-filter</code> <code>destination-prefix</code> <code>match-type <actions></code>	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 121.	You cannot specify this match condition.

Table 10: Routing Policy Match Conditions (*continued*)

Match Condition	Match Condition Category	from Statement Description	to Statement Description
route-type <i>value</i>	Standard	Type of route. The value can be one of the following: <ul style="list-style-type: none"> ■ external—External route. ■ internal—Internal route. 	
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 121.	You cannot specify this match condition.
state (active inactive)	Standard	(BGP export only) Match on the following types of advertised routes: <ul style="list-style-type: none"> ■ 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 <i>string</i> tag2 <i>string</i>	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. <p>You can specify multiple tags under one match condition by including the tags within a bracketed list. For example: from tag [tag1 tag2 tag3];</p> <p>For OSPF and IS-IS, the tag match conditions match the 32-bit tag field in external link-state advertisement (LSA) packets.</p>	

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 41.

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 20.

The following sections discuss the following actions:

- Configuring Flow Control Actions on page 48
- Configuring Actions That Manipulate Route Characteristics on page 49
- Configuring the Default Action in Routing Policies on page 54
- Configuring a Final Action in Routing Policies on page 56
- Logging Matches to a Routing Policy Term on page 56
- Configuring Separate Actions for Routes in Route Lists on page 57

Configuring Flow Control Actions

Table 11 on page 49 lists the flow control actions. You can specify one of these actions along with the trace action (see “Logging Matches to a Routing Policy Term”

on page 56) or one or more of the actions that manipulate route characteristics (see “Configuring Actions That Manipulate Route Characteristics” on page 49).

Table 11: 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	<p>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.</p> <p>next term is the default control action if a match occurs and you do not specify a flow control action.</p>
next policy	<p>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.</p> <p>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.</p>

Configuring Actions That Manipulate Route Characteristics

You can specify one or more of the actions listed in Table 12 on page 49 to manipulate route characteristics.

Table 12: Actions That Manipulate Route Characteristics

Action	Description
as-path-prepend <i>as-path</i>	<p>(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 within a confederation 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 137.</p> <p>In JUNOS Release 9.1 and later, you can specify 4-byte AS numbers as defined in RFC 4893, <i>BGP Support for Four-octet AS Number Space</i>, as well as the 2-byte AS numbers that are supported in earlier releases of the JUNOS Software. For more information about configuring AS numbers, see the <i>JUNOS Routing Protocols Configuration Guide</i>.</p>

Table 12: Actions That Manipulate Route Characteristics (*continued*)

Action	Description
as-path-expand last-as count <i>n</i>	(BGP only) Extract the last AS number in the existing AS path and affix that AS number to the beginning of the AS path <i>n</i> times, where <i>n</i> 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 <i>class-name</i>	(Class of service [CoS] only) Apply the specified class-of-service parameters to routes installed into the routing table. For more information, see the <i>JUNOS Class of Service Configuration Guide</i> .
color <i>preference</i> color2 <i>preference</i>	<p>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.</p> <p>If you set the preference with the color action, the value is internal to the JUNOS Software and is not transitive.</p> <p>For more information about preference values, see the <i>JUNOS Routing Protocols Configuration Guide</i>.</p>
color (add subtract) <i>number</i> color2 (add subtract) <i>number</i>	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) [<i>names</i>]	(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 104.
community (– delete) [<i>names</i>]	(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 104.
community (= set) [<i>names</i>]	(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 104.
cos-next-hop-map <i>map-name</i>	Set CoS-based next-hop map in forwarding table.
damping <i>name</i>	<p>(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.</p> <p>To apply damping parameters, you must enable BGP flap damping as described in the <i>JUNOS Routing Protocols Configuration Guide</i>, and you must create a named list of parameters as described in “Using Routing Policies to Damp BGP Route Flapping” on page 138.</p>

Table 12: Actions That Manipulate Route Characteristics (continued)

Action	Description
destination-class <i>destination-class-name</i>	<p>Maintain packet counts for a route passing through your network, based on the destination address in the packet. You can do the following:</p> <ul style="list-style-type: none"> ■ Configure group destination prefixes by configuring a routing policy; see “Defining Routing Policies” on page 40 and “Routing Policy Examples” on page 72. ■ Apply that routing policy to the forwarding table with the corresponding destination class; see “Applying Routing Policies to the Forwarding Table” on page 64. For more information about the forwarding-table configuration statement, see the <i>JUNOS Routing Protocols Configuration Guide</i>. ■ Enable packet counting on one or more interfaces by including the destination-class-usage statement at the [edit interfaces <i>interface-name</i> unit <i>logical-unit-number</i> family inet accounting] hierarchy level (see the <i>JUNOS Class of Service Configuration Guide</i>). See “Routing Policy Examples” on page 72. ■ View the output by using one of the following commands: show interfaces destination-class (all <i>destination-class-name logical-interface-name</i>), show interfaces interface-name extensive, or show interfaces interface-name statistics (see the <i>JUNOS Interfaces Command Reference</i>). ■ To configure a packet count based on the source address, use the source-class statement described in this table.
external type <i>metric</i>	Set the external metric type for routes exported by OSPF. You must specify the keyword type .
forwarding-class <i>forwarding-class-name</i>	<p>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:</p> <ul style="list-style-type: none"> ■ Configure group prefixes by configuring a routing policy; see “Defining Routing Policies” on page 40 and “Routing Policy Examples” on page 72. ■ Apply that routing policy to the forwarding table with the corresponding forwarding class; see “Applying Routing Policies to the Forwarding Table” on page 64. For more information about the forwarding-table configuration statement, see the <i>JUNOS Routing Protocols Configuration Guide</i>. ■ 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 <i>lsp-name</i>	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.
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 “Overview of Per-Packet Load Balancing” on page 144.
local-preference <i>value</i>	(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 ($2^{32} - 1$).

Table 12: Actions That Manipulate Route Characteristics (*continued*)

Action	Description
local-preference (add subtract) <i>number</i>	<p>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.</p> <p>For BGP, if the attribute value is not known, it is initialized to 100 before the routing policy is applied.</p>
map-to-interface (<i>interface-name</i> self)	<p>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:</p> <ul style="list-style-type: none"> 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. <p>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.</p> <ul style="list-style-type: none"> 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. <p>If no term matches, then no multicast data packets are sent.</p>
metric <i>metric</i> metric2 <i>metric</i> metric3 <i>metric</i> metric4 <i>metric</i>	<p>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.</p> <p>(BGP only) metric corresponds to the MED, and metric2 corresponds to the IGP metric if the BGP next hop loops through another router.</p>
metric (add subtract) <i>number</i> metric2 (add subtract) <i>number</i> metric3 (add subtract) <i>number</i> metric4 (add subtract) <i>number</i>	<p>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.</p>
metric expression (metric multiplier x offset a metric2 multiplier y offset b)	<p>Calculate a metric based on the current values of metric and metric2.</p> <p>This policy action overrides the current value of the metric attribute with the result of the expression</p> $((x * \text{metric}) + a) + ((y * \text{metric2}) + b)$ <p>where metric and metric2 are the current input values. Metric multipliers are limited in range to eight significant digits.</p>
metric (igp minimum-igp) site-offset	<p>(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.</p>

Table 12: Actions That Manipulate Route Characteristics (*continued*)

Action	Description
next-hop (<i>address</i> <i>discard</i> next-table <i>routing-table-name</i> peer-address <i>reject</i> <i>self</i>)	<p>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.</p> <p>If you specify self, the next-hop address is replaced by one of the local router'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 router cannot install routes with itself as the next hop.</p> <p>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 router or another directly connected router.</p> <p>If you specify discard, the next-hop address is replaced by a discard next hop.</p> <p>If you specify next-table, the router performs a forwarding lookup in the specified table.</p> <p>If you specify reject, the next-hop address is replaced by a reject next hop.</p>
origin <i>value</i>	<p>(BGP only) Set the BGP origin attribute to one of the following values:</p> <ul style="list-style-type: none"> ■ igp—Path information originated within the local AS. ■ egp—Path information originated in another AS. ■ incomplete—Path information learned by some other means.
preference <i>preference</i> preference2 <i>preference</i>	<p>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.</p> <p>To specify even finer-grained preference values, see the color and color2 actions in this table.</p> <p>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 Software and is not transitive.</p> <p>For more information about preference values, see the <i>JUNOS Routing Protocols Configuration Guide</i>.</p>
preference (add subtract) <i>number</i> preference2 (add subtract) <i>number</i>	<p>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.</p>
priority (low medium high)	<p>(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. For more detailed information about configuring priority for prefixes included in OSPF import policy, see the <i>JUNOS Routing Protocols Configuration Guide</i>.</p> <p>NOTE: 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.</p>

Table 12: Actions That Manipulate Route Characteristics (*continued*)

Action	Description
<code>source-class source-class-name</code>	<p>Maintain packet counts for a route passing through your network, based on the source address. You can do the following:</p> <ul style="list-style-type: none"> ■ Configure group source prefixes by configuring a routing policy; see “Defining Routing Policies” on page 40 and “Routing Policy Examples” on page 72. ■ Apply that routing policy to the forwarding table with the corresponding source class; see “Applying Routing Policies to the Forwarding Table” on page 64. For more information about the <code>forwarding-table</code> configuration statement, see the <i>JUNOS Routing Protocols Configuration Guide</i>. ■ Enable packet counting on one or more interfaces by including the <code>source-class-usage interface-name</code> statement at the <code>[edit interfaces logical-unit-number unit family inet accounting]</code> hierarchy level (see the <i>JUNOS Network Interfaces Configuration Guide</i>). Also, follow the <code>source-class-usage</code> statement with the <code>input</code> or <code>output</code> 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 <code>[edit interfaces interface-name unit family inet accounting source-class-usage (input output input output) unit-number]</code>. See the example in “Routing Policy Examples” on page 72. ■ View the output by using one of the following commands: <code>show interfaces interface-name source-class source-class-name</code>, <code>show interfaces interface-name extensive</code>, or <code>show interfaces interface-name statistics</code> (see the <i>JUNOS Interfaces Command Reference</i>). ■ To configure a packet count based on the destination address, use the <code>destination-class</code> statement described in this table. ■ For a detailed source-class usage example configuration, see the <i>JUNOS Feature Guide</i>.
<code>tag tag tag2 tag</code>	<p>Set the tag value. You can specify two tag strings: <code>tag</code> (for the first string) and <code>tag2</code> (a second string). These values are local to the router.</p> <ul style="list-style-type: none"> ■ For OSPF routes the <code>tag</code> action sets the 32-bit tag field in OSPF external link-state advertisement (LSA) packets. ■ For IS-IS routes, the <code>tag</code> action sets the 32-bit flag in the IS-IS IP prefix type length values (TLV). ■ For RIPv2 routes, the <code>tag</code> action sets the route-tag community. The <code>tag2</code> option is not supported.
<code>tag (add subtract) number tag2 (add subtract) number</code>	<p>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.</p>

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

For more information about the global **traceoptions** statement, see the *JUNOS Routing Protocols Configuration Guide*.

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 121.

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. Table 5 on page 18 summarizes the import and export policy support for each routing protocol and the level at which you can apply these policies.

For more information about applying routing policies to individual routing protocols, see the *JUNOS Routing Protocols Configuration Guide*.

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 29.

Effect of Omitting Ingress Match Conditions from Export Policies

In export policies, omitting the **from** statement in a term might lead to unexpected results. By default, if you omit the **from** statement, all routes are considered to match. For example, static and direct routes are not exported by BGP by default. However, if you create a term with an empty **from** statement, these routes inadvertently could be exported because they matched the **from** statement. For example, the following routing policy is designed to reject a few route ranges and then export routes learned by BGP (which is the default export behavior):

```
[edit]
routing-options {
  autonomous-system 56;
}
protocols {
  bgp {
    group 4 {
      export statics-policy;
      type external;
      peer-as 47;
      neighbor 192.168.1.1;
    }
  }
}
policy-options {
  policy-statement statics-policy {
    term term1 {
      from {
        route-filter 192.168.0.0/16 orlonger;
        route-filter 172.16.1.1/3 orlonger;
      }
      then reject; # reject the prefixes in the route list
    }
    term term2 {
      then {
        accept; # accept all other routes, including static and direct routes
      }
    }
  }
}
```

However, this routing policy results in BGP advertising static and direct routes to its peers because:

- **term1** rejects the destination prefixes enumerated in the route list.

- **term2**, because it has no **from** statement, matches all other routes, including static and direct routes, and accepts all these routes (with the **accept** statement).

To modify the preceding routing policy so that an IGP does not export unwanted routes, you can specify the following additional terms:

```
[edit]
routing-options {
  autonomous-system 56;
}
protocols {
  isis {
    export statics-policy;
  }
}
policy-options {
  policy-statement statics-policy {
    term term1 {
      from {
        route-filter 192.168.0.0/16 orlonger;
        route-filter 172.16.1.1/3 orlonger;
      }
      then reject; # reject the prefixes in the route list
    }
    term term2 { # reject direct routes
      from protocol direct;
      then reject;
    }
    term term3 { # reject static routes
      from protocol static;
      then reject;
    }
    term term4 { # reject local routes
      from protocol local;
      then reject;
    }
    term term5 { # reject aggregate routes
      from protocol aggregate;
      then reject;
    }
    term term6 {
      then accept; # accept all other routes
    }
  }
}
```

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 13 on page 60 summarizes the policy actions and their corresponding TRUE and FALSE values and flow control action values. Table 14 on page 60 describes the logical operators. For complete information about policy expression evaluation, see “How a Policy Expression Is Evaluated” on page 62.

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 13: 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 14: Policy Expression Logical Operators

Logical Operator	Policy Expression Logic	How Logical Operator Affects Policy Expression Evaluation
&& (Logical AND)	<p>Logical AND requires that all values must be TRUE to produce output of TRUE.</p> <p>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.</p>	<p>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.</p>
(Logical OR)	<p>Logical OR requires that at least one value must be TRUE to produce output of TRUE.</p> <p>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.</p>	<p>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.</p>

Table 14: Policy Expression Logical Operators *(continued)*

Logical Operator	Policy Expression Logic	How Logical Operator Affects Policy Expression Evaluation
! (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.	<p>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.</p> <p>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.</p> <p>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.</p>

For more information, see the following sections:

- Policy Expression Examples on page 61
- How a Policy Expression Is Evaluated on page 62
- Example: Evaluating Policy Expressions on page 63

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 62.) 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 13 on page 60.)
2. The software takes the value of TRUE or FALSE and evaluates it against the logical operator used in the policy expression (see Table 14 on page 60). 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 13 on page 60.) 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, **accept**, 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 29.

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 “Overview of Per-Packet Load Balancing” on page 144. For more information about CoS, see the *JUNOS 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 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 Software 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 on page 66
- Configuring Routing Policies Based on Dynamic Database Configuration on page 67

- Applying Dynamic Routing Policies to BGP on page 68
- Preventing Reestablishment of BGP Peering Sessions After NSR Routing Engine Switchover on page 68
- Example: Configuring a BGP Export Policy That References a Dynamic Routing Policy on page 69

Configuring Routing Policies and Policy Objects in the Dynamic Database

JUNOS 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 Software to JUNOS 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. For more general information about configuring BGP import and export policy, see the *JUNOS Routing Protocols Configuration Guide*.

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 BGP and the configuration statement summary of the `idle-after-switch-over` statement, see the *JUNOS Routing Protocols Configuration Guide*. For more information about configuring active nonstop routing, see the *JUNOS 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. For more information about configuring interfaces, see the *JUNOS Network Interfaces Configuration Guide*.

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
```

For more information about test commands, see the *JUNOS Routing Protocols and Policies Command Reference*.

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:

- Example: Defining a Routing Policy from BGP to IS-IS on page 73
- Example: Using Routing Policy to Set a Preference on page 74
- Example: Importing and Exporting Access and Access-Internal Routes in a Routing Policy on page 74

- Example: Exporting Routes to IS-IS on page 75
- Example: Applying Export and Import Policies to BGP Peer Groups on page 75
- Example: Applying a Prefix to Routes Learned from a Peer on page 76
- Example: Redistributing BGP Routes with a Specific Community Tag into IS-IS on page 76
- Example: Redistributing OSPF Routes into BGP on page 76
- Example: Exporting Direct Routes Into IS-IS on page 77
- Example: Exporting Internal IS-IS Level 1 Routes to Level 2 on page 77
- Example: Exporting IS-IS Level 2 Routes to Level 1 on page 78
- Example: Assigning Different Forwarding Next-Hop LSPs to Different Destination Prefixes on page 78
- Example: Grouping Destination Prefixes on page 79
- Example: Grouping Source Prefixes on page 80
- Example: Grouping Source and Destination Prefixes in a Forwarding Class on page 81
- Example: Accepting Routes with Specific Destination Prefixes on page 82
- Example: Accepting Routes from BGP with a Specific Destination Prefix on page 83

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 20.) 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 20.)

```
[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 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: 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.

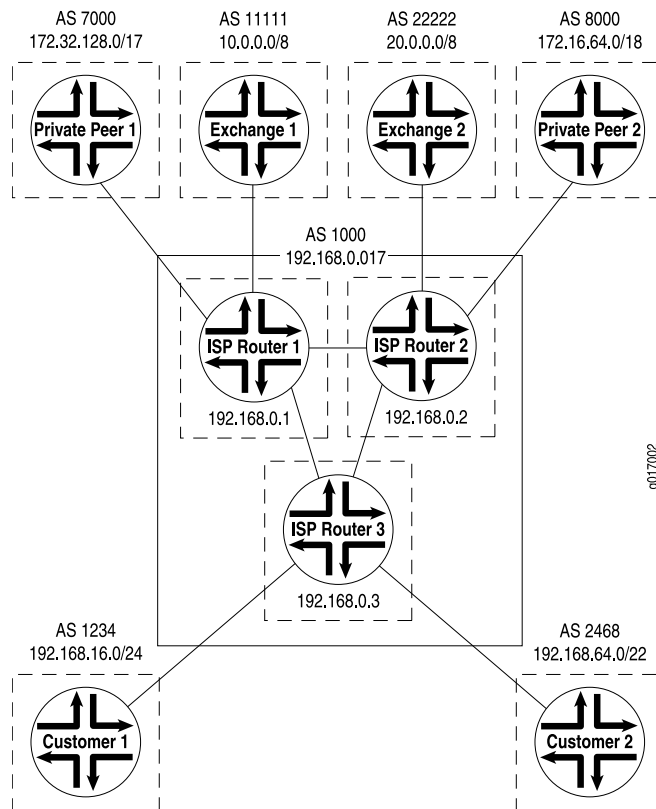
```

[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 11 on page 84), 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 11: 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 11 on page 84, 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 on page 85
- Requesting Specific Routes on the Customer 2 Router on page 86
- Configuring a Peer Policy on ISP Router 3 on page 88
- Configuring Private and Exchange Peers on ISP Router 1 and 2 on page 90
- Configuring Locally Defined Static Routes on the Exchange Peer 2 Router on page 93
- Configuring Outbound and Generated Routes on the Private Peer 2 Router on page 93

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 5

Extended Match Conditions Configuration

This chapter describes how to configure extended match conditions:

- Configuring AS Path Regular Expressions to Use as Routing Policy Match Conditions on page 97
- Overview of BGP Communities and Extended Communities as Routing Policy Match Conditions on page 104
- Defining BGP Communities and Extended Communities for Use in Routing Policy Match Conditions on page 106
- Including BGP Communities and Extended Communities in Routing Policy Match Conditions on page 111
- How BGP Communities and Extended Communities Are Evaluated in Routing Policy Match Conditions on page 112
- Using Routing Policies to Prevent Advertisement of BGP Communities to Neighbors on page 113
- Examples: Configuring BGP Communities as Routing Policy Match Conditions on page 113
- Configuring Prefix Lists for Use in Routing Policy Match Conditions on page 117
- Configuring Route Lists for Use in Routing Policy Match Conditions on page 121
- Configuring Subroutines in Routing Policy Match Conditions on page 130
- Configuring Routing Policy Match Conditions Based on Routing Table Entries on page 134

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 on page 98
- How AS Path Regular Expressions Are Evaluated on page 103
- Examples: Configuring AS Path Regular Expressions on page 103

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 Table 10 on page 42). 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 255 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 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 Software. You can configure a value in the range from 1 through 4,294,967,295. For more information about configuring AS numbers, see the *JUNOS Routing Protocols Configuration Guide*.

- **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 15 on page 100 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 16 on page 100 shows examples of how to define regular expressions to match AS paths.

Table 15: AS Path Regular Expression Operators

Operator	Match Definition
{ <i>m</i> , <i>n</i> }	At least <i>m</i> and at most <i>n</i> repetitions of <i>term</i> . Both <i>m</i> and <i>n</i> must be positive integers, and <i>m</i> must be smaller than <i>n</i> .
{ <i>m</i> }	Exactly <i>m</i> repetitions of <i>term</i> . <i>m</i> must be a positive integer.
{ <i>m</i> ,}	<i>m</i> or more repetitions of <i>term</i> . <i>m</i> must be a positive integer.
*	Zero or more repetitions of <i>term</i> . This is equivalent to {0,}.
+	One or more repetitions of <i>term</i> . This is equivalent to {1,}.
?	Zero or one repetition of <i>term</i> . 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 16: Examples of AS Path Regular Expressions

AS Path to Match	Regular Expression	Sample Matches
AS path is 1234	1234	1234

Table 16: Examples of AS Path Regular Expressions *(continued)*

AS Path to Match	Regular Expression	Sample Matches
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
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
Path whose second AS number must be 56 or 78	(. 56) (. 78) or . (56 78)	1234 56
		1234 78
		9876 56
		3857 78

Table 16: Examples of AS Path Regular Expressions *(continued)*

AS Path to Match	Regular Expression	Sample Matches
Path whose second AS number might be 56 or 78	<code>.(56 78)?</code>	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	<code>123 (56 78)</code>	123 56 123 78
Path of any length, except nonexistent, whose second AS number can be anything, including nonexistent	<code>..* or ..{0,}</code>	12341234567812345678
AS path is 1 2 3	<code>1 2 3</code>	1 2 3
One occurrence of the AS numbers 1 and 2, followed by one or more occurrences of the AS number 3	<code>1 2 3 +</code>	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	<code>1 + 2 + 3 +</code>	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	<code>4 5 6 .*</code>	4 5 6 4 5 6 7 8 9
Path of any length that ends with AS numbers 4, 5, 6	<code>.* 4 5 6</code>	4 5 6 1 2 3 4 5 6 4 9 4 5 6
AS path 5, 12, or 18	<code>5 12 18</code>	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;
  }
}

```

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 Table 10 on page 42. For a list of the actions that can be configured for communities, see Table 12 on page 49.

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 Software 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 106
- Including BGP Communities and Extended Communities in Routing Policy Match Conditions on page 111
- How BGP Communities and Extended Communities Are Evaluated in Routing Policy Match Conditions on page 112
- Using Routing Policies to Prevent Advertisement of BGP Communities to Neighbors on page 113
- Examples: Configuring BGP Communities as Routing Policy Match Conditions on page 113

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

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 on page 106
- Defining BGP Extended Communities for Use in Routing Policy Match Conditions on page 109
- Inverting Community Matches on page 111

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. For more information about configuring AS numbers, see the *JUNOS Routing Protocols Configuration Guide*. 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 17 on page 108.)
 - 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 107.

- **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 17 on page 108.)
 - 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 17 on page 108 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 18 on page 109 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 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 Software. For more information about configuring AS numbers, see the *JUNOS Routing Protocols Configuration Guide*.

Table 17: Community Attribute Regular Expression Operators

Operator	Match Definition
$\{m,n\}$	At least <i>m</i> and at most <i>n</i> repetitions of <i>term</i> . Both <i>m</i> and <i>n</i> must be positive integers, and <i>m</i> must be smaller than <i>n</i> .
$\{m\}$	Exactly <i>m</i> repetitions of <i>term</i> . <i>m</i> must be a positive integer.
$\{m,\}$	<i>m</i> or more repetitions of <i>term</i> . <i>m</i> must be a positive integer.
*	Zero or more repetitions of <i>term</i> . This is equivalent to $\{0,\}$.
+	One or more repetitions of <i>term</i> . This is equivalent to $\{1,\}$.
?	Zero or one repetition of <i>term</i> . 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.

Table 17: Community Attribute Regular Expression Operators (*continued*)

Operator	Match Definition
" "	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 18: 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.

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 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 Software. 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 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.

For more information about configuring AS numbers, see the *JUNOS Routing Protocols Configuration Guide*.

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 ];
```

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).

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 107). 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 ];
}
```



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

- 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**.

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 wild** 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 `lcl20x-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 `lcl20x-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 `internal20x` 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 `prepend20x-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 `peer20x`, `custpeer20x`, and `internal20x` 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 19 on page 118.

Table 19: 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 121.

This section includes the following information:

- Configuring Prefix Lists on page 118
- How Prefix Lists Are Evaluated in Routing Policy Match Conditions on page 119
- Configuring Prefix List Filters on page 120
- Example: Configuring a Prefix List on page 120

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 Table 10 on page 42).

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 *Configuring 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 120; for examples of configuring a firewall filter, see *Configuring Standard Firewall Filters*.

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” on page 117.

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 20 on page 120.

The *actions* option is the action to take if the prefix list matches. It can be one or more of the actions listed in Table 11 on page 49 and Table 12 on page 49.

Table 20: Route List Match Types for a Prefix List Filter

Match Type	Match Condition
exact	The route shares the same most-significant bits (described by <i>prefix-length</i>), and <i>prefix-length</i> is equal to the route's prefix length.
longer	The route shares the same most-significant bits (described by <i>prefix-length</i>), and <i>prefix-length</i> is greater than the route's prefix length.
orlonger	The route shares the same most-significant bits (described by <i>prefix-length</i>), and <i>prefix-length</i> 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 destination prefixes. When specifying a 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 124 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 on page 122
- How Route Lists Are Evaluated in Routing Policy Match Conditions on page 124
- Route List Examples on page 126

Configuring Route Lists

To configure a route list, include one or more `route-filter` or `source-address-filter` statements 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 prefix match-type {
    action;
}
source-address-filter source-prefix match-type {
    action;
}
```

The `route-filter` option is typically used to match prefixes of any type except for unicast source addresses.

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

`source-prefix` is the IPv4 or IPv6 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 destination prefix. It can be one of the match types listed in Table 21 on page 123. For examples of the match types and the results when presented with various routes, see Table 22 on page 123.

`actions` is the action to take if the destination prefix matches. It can be one or more of the actions listed in Table 11 on page 49 and Table 12 on page 49.

In route lists, 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 and if an action is not specified in 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 126.

Table 21: Route List Match Types for a Prefix List

Match Type	Match Condition
exact	The route shares the same most-significant bits (described by <i>prefix-length</i>), and <i>prefix-length</i> is equal to the route’s prefix length.
longer	The route shares the same most-significant bits (described by <i>prefix-length</i>), and <i>prefix-length</i> is greater than the route’s prefix length.
orlonger	The route shares the same most-significant bits (described by <i>prefix-length</i>), and <i>prefix-length</i> is equal to or greater than the route’s prefix length.
prefix-length-range <i>prefix-length2-prefix-length3</i>	The route shares the same most-significant bits (described by <i>prefix-length</i>), and the route’s prefix length falls between <i>prefix-length2</i> and <i>prefix-length3</i> , inclusive.
through <i>destination-prefix</i>	<p>All the following are true:</p> <ul style="list-style-type: none"> ■ The route shares the same most-significant bits (described by <i>prefix-length</i>) of the first destination prefix. ■ The route shares the same most-significant bits (described by <i>prefix-length</i>) of the second destination prefix for the number of bits in the prefix length. ■ The number of bits in the route’s prefix length is less than or equal to the number of bits in the second prefix. <p>You do not use the through match type in most routing policy configurations. (For an example, see “Route List Examples” on page 126.)</p>
upto <i>prefix-length2</i>	The route shares the same most-significant bits (described by <i>prefix-length</i>) and the route’s prefix length falls between <i>prefix-length</i> and <i>prefix-length2</i> .

Table 22: Match Type Examples

Prefix	192.168/16 exact	192.168/16 longer	192.168/16 orlonger	192.168/16 upto /24	192.168/16 through 192.168.16/20	192.168/16 prefix length range /18–/20
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
192.168.4.0/24	–	Match	Match	Match	–	–
192.168.5.4/30	–	Match	Match	–	–	–
192.168.12.4/30	–	Match	Match	–	–	–

Table 22: 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 through 192.168.16/20	192.168/16 prefix length range /18-/20
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
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 only and not the match type. 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 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 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.

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 20.



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 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 20.) The shorter prefix **192.168.0.0/16** or longer that you wanted to match is inadvertently ignored.

One solution to this problem is to remove the prefix **192.168.0.0/16** or longer from the route list in this term and move it to a previous 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 20.

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

- Example: Rejecting Routes with Specific Destination Prefixes and Mask Lengths on page 126
- Example: Rejecting Routes with a Mask Length Greater than Eight on page 127
- Example: Rejecting Routes with Mask Length Between 26 and 29 on page 127
- Example: Rejecting Routes from Specific Hosts on page 127
- Example: Accepting Routes with a Defined Set of Prefixes on page 128
- Example: Rejecting Routes with a Defined Set of Prefixes on page 128
- Example: Rejecting Routes with Prefixes Longer than 24 Bits on page 129
- Example: Rejecting PIM Multicast Traffic Joins on page 129
- Example: Rejecting PIM Traffic on page 130

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;
        route-filter 0/0;
      }
      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 Multicast Protocols Configuration Guide*.

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 130
- Example: Configuring a Subroutine on page 134

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 Table 10 on page 42) in the **from** or **to** statement of another routing policy:

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

```

        prefix-list name;
    }
    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 31.

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 20.)

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. (Also see “Effect of Omitting Ingress Match Conditions from Export Policies” on page 58.)
- 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; for more information, see the *JUNOS Routing Protocols and Policies Command Reference*.

Chapter 6

Extended Actions Configuration

This chapter provides information about the following routing policy actions configuration tasks:

- Prepending AS Numbers to BGP AS Paths on page 137
- Adding AS Numbers to BGP AS Paths on page 138
- Using Routing Policies to Damp BGP Route Flapping on page 138
- Overview of Per-Packet Load Balancing on page 144
- Configuring Per-Packet Load Balancing on page 145
- Configuring Load Balancing Based on MPLS Labels on page 147
- Configuring Load Balancing for Ethernet Pseudowires on page 150
- Configuring Load Balancing Based on MAC Addresses on page 151
- Configuring VPLS Load Balancing Based on IP and MPLS Information on page 151
- Configuring VPLS Load Balancing on MX Series Ethernet Services Routers on page 153

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 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 Software. In plain-number format, you can configure a value in the range from 1 through 4,294,967,295. For more information about configuring AS numbers, see the *JUNOS Routing Protocols Configuration Guide*.

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";
  }
}

```

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 Table 12 on page 49). For the damping routing policy to work, you also must enable BGP route flap damping.

For further information about enabling BGP route flap damping, see the *JUNOS Routing Protocols Configuration Guide*.

The following sections discuss the following topics:

- Configuring BGP Flap Damping Parameters on page 139
- Specifying BGP Flap Damping as the Action in Routing Policy Terms on page 141
- Disabling Damping for Specific Address Prefixes on page 142
- Example: Configuring BGP Flap Damping on page 142

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 23 on page 140.

Table 23: Damping Parameters

Damping Parameter	Description	Default	Possible Values
half-life <i>minutes</i>	Decay half-life, in minutes	15 minutes	1 through 45 minutes
max-suppress <i>minutes</i>	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 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.

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.

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:

$$\epsilon_c \leq \epsilon_r e^{(t/\lambda) (\ln 2)}$$

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:

$$\epsilon_c \leq 750 e^{(60/30) (\ln 2)}$$

$$\epsilon_c \leq 3000$$



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 router, 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 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

```

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 Software 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 Software 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.



NOTE: You can configure per-packet load balancing to optimize VPLS traffic flows across multiple paths.

For information about configuring per-packet load balancing, see the following topics:

- Configuring Per-Packet Load Balancing on page 145
- Configuring Load Balancing Based on MPLS Labels on page 147
- Configuring Load Balancing for Ethernet Pseudowires on page 150
- Configuring Load Balancing Based on MAC Addresses on page 151

- Configuring VPLS Load Balancing Based on IP and MPLS Information on page 151
- Configuring VPLS Load Balancing on MX Series Ethernet Services Routers on page 153

Configuring Per-Packet Load Balancing

To configure per-packet load balancing as described in “Overview of Per-Packet Load Balancing” on page 144, 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 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; for more information about this action, see “Routing Policy Configuration” on page 39.

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

To load-balance based on the MPLS label information, include the `family mpls` statement at the `[edit forwarding-options hash-key]` hierarchy level:

```
[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;
    }
  }
}

```

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. For more information about configuring load balancing for Ethernet pseudowires, see “Configuring Load Balancing for Ethernet Pseudowires” on page 150.

To include the first label in the hash key, include the **label-1** option. This is used for a one-label packet.

To include the first and second label in the hash key, include both the **label-1** and **label-2** options. This is used for a two-label packet. The router provides hashing on the first and second labels by default. If both labels are specified, the entire first label and the first 16 bits of the second label are hashed.

To include the third MPLS label in the hash function, include the **label-3** option at the **[edit forwarding-options hash-key family mpls]** hierarchy level. To include no MPLS labels in the hash function, include the **no-labels** option at the **[edit forwarding-options hash-key family mpls]** hierarchy level.

Hashing can include IP addresses to provide better distribution of traffic to aggregated interfaces.

To include the bits in the IP address of the IPv4 or IPv6 payload as well as the first label in the hash key, include the **label-1** and **payload ip** statements 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 bits of the IP address of the IPv4 or IPv6 payload as well as both the first label and the second label in the hash key, include the **label-1**, **label-2**, and **payload ip** statements at the **[edit forwarding-options hash-key family mpls]** hierarchy level.

```
[edit forwarding-options hash-key family mpls]
label-1;
  label-2;
  label-3;
  no-labels;
  payload {
    ip {
      layer-3-only;
      port-data {
        source-msb;
        source-lsb;
        destination-msb;
        destination-lsb;
      }
    }
  }
}
```

To include only Layer 3 IP information in the hash key, specify the **layer-3-only** option at the `[edit forwarding-options hash-key family mpls payload ip]` hierarchy level. To include the most significant byte of the source port, specify the **source-msb** option. To include the least significant byte of the source port, specify the **source-lsb** option. To include the most significant byte of the destination port, specify the **destination-msb** option. To include the least significant byte of the destination port, specify the **destination-lsb** option.

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** option at the `[edit forwarding-options hash-key family mpls payload ip]` hierarchy level.

In an Layer 2 VPN scenario, the router could encounter a reordering complication. When a burst of traffic pushes the customer traffic bandwidth to exceed its limits, the traffic might be affected in mid flow. Packets may be reordered as a result.

You can configure the EXP bit of the top label to be excluded from the hash calculations to avoid the reordering complication. 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;
  }
}
```

You must also configure the **label-1** statement when configuring the **no-label-1-exp** statement.

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 M1 20, 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

To load-balance traffic based on Layer 2 media access control (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: You can configure per-packet load balancing to optimize VPLS traffic flows across multiple paths. For more detailed information, see the *JUNOS VPNS Configuration Guide*.



NOTE: J Series Services Routers do not support this feature.

Configuring VPLS Load Balancing Based on IP and MPLS Information

In JUNOS 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 releases, you can configure load balancing based only on Layer 2 information. In JUNOS 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 Ethernet Services Routers” on page 153.

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 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 Ethernet Services Routers

In JUNOS 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 151.

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 Topics**
- `family multiservice`
 - `hash-key`

Chapter 7

Summary of Routing Policy Configuration Statements

The following sections explain each of the routing policy configuration statements. The statements are organized alphabetically.

apply-path

Syntax	<code>apply-path path;</code>
Hierarchy Level	[edit logical-systems <i>logical-system-name</i> policy-options prefix-list <i>name</i>], [edit policy-options prefix-list <i>name</i>]
Release Information	Statement introduced before JUNOS Release 7.4.
Description	Expand a prefix list to include all prefixes pointed to by a defined path.
Options	<p><i>path</i>—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.</p> <p><i>prefix-list name</i>—Name of a list of IP version 4 (IPv4) or IP version 6 (IPv6) prefixes. To create a named list of IP address prefixes, see “Extended Match Conditions Configuration” on page 97.</p>
Usage Guidelines	See “Configuring Prefix Lists” on page 118.
Required Privilege Level	routing—To view this statement in the configuration. routing-control—To add this statement to the configuration.

as-path

Syntax	<code>as-path name regular-expression;</code>
Hierarchy Level	[edit dynamic policy-options], [edit logical-systems <i>logical-system-name</i> policy-options], [edit policy-options]
Release Information	Statement introduced before JUNOS Release 7.4. Support for configuration in the dynamic database introduced in JUNOS Release 9.5.
Description	Define an autonomous system (AS) path regular expression for use in a routing policy match condition.
Options	<i>name</i> —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 (" "). <i>regular-expression</i> —One or more regular expressions used to match the AS path.
Usage Guidelines	See “Configuring AS Path Regular Expressions to Use as Routing Policy Match Conditions” on page 97 and “Configuring Routing Policies and Policy Objects in the Dynamic Database” on page 66.



NOTE: Because the JUNOS Software 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.

Required Privilege Level	routing—To view this statement in the configuration. routing-control—To add this statement to the configuration.
Related Topics	dynamic-db

as-path-group

Syntax	as-path-group <i>group-name</i> { as-path <i>name</i> <i>regular-expression</i> ; }
Hierarchy Level	[edit dynamic policy-options], [edit logical-systems <i>logical-system-name</i> policy-options], [edit policy-options]
Release Information	Statement introduced before JUNOS Release 7.4. Support for dynamic database configuration introduced in JUNOS Release 9.5.
Description	Define a group containing multiple AS path regular expressions for use in a routing policy match condition.
Options	<p><i>group-name</i>—Name that identifies the AS path group. One or more AS path regular expressions must be listed below the as-path-group hierarchy.</p> <p><i>name</i>—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 (" ").</p> <p><i>regular-expression</i>—One or more regular expressions used to match the AS path.</p>
Usage Guidelines	See “Configuring AS Path Regular Expressions to Use as Routing Policy Match Conditions” on page 97 and “Configuring Routing Policies and Policy Objects in the Dynamic Database” on page 66.



NOTE: Because the JUNOS Software 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.

Required Privilege Level	routing—To view this statement in the configuration. routing-control—To add this statement to the configuration.
Related Topics	dynamic-db

community

Syntax	<pre>community <i>name</i> { invert-match; members [<i>community-ids</i>]; }</pre>
Hierarchy Level	[edit dynamic policy-options], [edit logical-systems <i>logical-system-name</i> policy-options], [edit policy-options]
Release Information	Statement introduced before JUNOS Release 7.4. Support for configuration in the dynamic database introduced in JUNOS Release 9.5.
Description	Define a community or extended community for use in a routing policy match condition.
Options	<p><i>name</i>—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 (" ").</p> <p><i>invert-match</i>—Invert the results of the community expression matching.</p> <p><i>members community-ids</i>—One or more community members. If you specify more than one member, you must enclose all members in brackets.</p> <p>The format for <i>community-ids</i> is:</p> <pre><i>as-number:community-value</i></pre> <p><i>as-number</i> is the AS number and can be a value in the range from 0 through 65,535. <i>community-value</i> is the community identifier and can be a number in the range from 0 through 65,535.</p> <p>You also can specify <i>community-ids</i> for communities as one of the following well-known community names, which are defined in RFC 1997, <i>BGP Communities Attribute</i>:</p> <ul style="list-style-type: none"> ■ 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. <p>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.</p>

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 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 Software. 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 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.

For more information about configuring AS numbers, see the *JUNOS Routing Protocols Configuration Guide*.

Usage Guidelines See “Overview of BGP Communities and Extended Communities as Routing Policy Match Conditions” on page 104, “Defining BGP Communities and Extended Communities for Use in Routing Policy Match Conditions” on page 106, and “Configuring Routing Policies and Policy Objects in the Dynamic Database” on page 66.



NOTE: Because the JUNOS Software 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.

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

Related Topics dynamic-db

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 Release 9.0.
Support for configuration in the dynamic database introduced in JUNOS Release 9.5.

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.

Usage Guidelines See “Configuring Routing Policy Match Conditions Based on Routing Table Entries” on page 134 and “Configuring Routing Policies and Policy Objects in the Dynamic Database” on page 66.



NOTE: Because the JUNOS Software 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.

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

Related Topics dynamic-db

damping

Syntax	<pre>damping <i>name</i> { disable; half-life <i>minutes</i>; max-suppress <i>minutes</i>; reuse <i>number</i>; suppress <i>number</i>; }</pre>
Hierarchy Level	[edit logical-systems <i>logical-system-name</i> policy-options], [edit policy-options]
Release Information	Statement introduced before JUNOS Release 7.4.
Description	Define route flap damping properties to set on BGP routes.
Options	<p>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.</p> <p>half-life <i>minutes</i>—Decay half-life. <i>minutes</i> 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</p> <p>max-suppress <i>minutes</i>—Maximum hold-down time. <i>minutes</i> is the maximum time that a route can be suppressed no matter how unstable it has been. Range: 1 through 720 Default: 60 minutes</p> <p><i>name</i>—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 (" ").</p> <p>reuse <i>number</i>—Reuse threshold. <i>number</i> is the figure-of-merit value below which a suppressed route can be used again. Range: 1 through 20,000 Default: 750 (unitless)</p> <p>suppress <i>number</i>—Cutoff (suppression) threshold. <i>number</i> 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)</p>
Usage Guidelines	See “Configuring BGP Flap Damping Parameters” on page 139.
Required Privilege Level	<p>routing—To view this statement in the configuration.</p> <p>routing-control—To add this statement to the configuration.</p>

dynamic-db

Syntax	dynamic-db;
Hierarchy Level	[edit logical-systems <i>logical-system-name</i> policy-options as-path <i>path-name</i>], [edit logical-systems <i>logical-system-name</i> policy-options as-path-group <i>group-name</i>], [edit logical-systems <i>logical-system-name</i> policy-options community <i>community-name</i>], [edit logical-systems <i>logical-system-name</i> policy-options condition <i>condition-name</i>], [edit logical-systems <i>logical-system-name</i> policy-options policy-statement <i>policy-statement-name</i>], [edit logical-systems <i>logical-system-name</i> policy-options prefix-list <i>prefix-list-name</i>], [edit policy-options as-path <i>path-name</i>], [edit policy-options as-path-group <i>group-name</i>], [edit policy-options community <i>community-name</i>], [edit policy-options condition <i>condition-name</i>], [edit policy-options policy-statement <i>policy-statement-name</i>], [edit policy-options prefix-list <i>prefix-list-name</i>]
Release Information	Statement introduced in JUNOS Release 9.5.
Description	Define routing policies and policy objects that reference policies configured in the dynamic database at the [edit dynamic] hierarchy level.
Usage Guidelines	See “Configuring Routing Policies Based on Dynamic Database Configuration” on page 67.
Required Privilege Level	routing—To view this statement in the configuration. routing-control-level—To add this statement to the configuration.

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 Release 7.4.
Description	Apply one or more policies to routes being exported from the routing table into a routing protocol.
Options	<i>policy-names</i> —Names of one or more policies defined with a <i>policy-statement</i> statement.
Usage Guidelines	See “Applying Routing Policies and Policy Chains to Routing Protocols” on page 57.
Required Privilege Level	routing—To view this statement in the configuration. routing-control—To add this statement to the configuration.

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 Release 7.4.
Description	Apply one or more policies to routes being imported into the routing table from a routing protocol.
Options	<i>policy-names</i> —Names of one or more policies defined with a <i>policy-statement</i> statement.
Usage Guidelines	See “Applying Routing Policies and Policy Chains to Routing Protocols” on page 57.
Required Privilege Level	routing—To view this statement in the configuration. routing-control—To add this statement to the configuration.

policy-options

Syntax	<code>policy-options { ... }</code>
Hierarchy Level	[edit], [edit dynamic]
Release Information	Statement introduced before JUNOS Release 7.4.
Description	Configure routing policy.
Options	The statements are explained separately.
Usage Guidelines	See “Defining Routing Policies” on page 40.
Required Privilege Level	routing—To view this statement in the configuration. routing-control—To add this statement to the configuration.

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 Release 7.4.
Support for configuration in the dynamic database introduced in JUNOS Release 9.5.

Description Define a routing policy, including subroutine policies.

Options *actions*—(Optional) One or more actions to take if the conditions match. The actions are described in Table 11 on page 49 and Table 12 on page 49.

family family-name—(Optional) Specify an address family protocol. Specify *inet* for an IPv4 address protocol. Specify *inet6* for a 128-bit IPv6 address protocol, and to enable interpretation of IPv6 router filter addresses. For IS-IS traffic, For IPv4 multicast VPN traffic, specify *inet-mvpn*. For IPv6 multicast VPN traffic, specify *inet6-mvpn*.



NOTE: When *family* is not specified, the router 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 Table 10 on page 42.

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 130.

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. To create a named list of IP address prefixes, see “Extended Match Conditions Configuration” on page 97.

prefix-list-filter prefix-list-name—Name of a prefix list to evaluate using qualifiers; **match-type** is the type of match (see Table 21 on page 123), 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 Table 20 on page 120), 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 Table 21 on page 123), 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 Table 11 on page 49 and Table 12 on page 49.

Usage Guidelines See “Defining Routing Policies” on page 40, “Extended Match Conditions Configuration” on page 97 and “Configuring Routing Policies and Policy Objects in the Dynamic Database” on page 66.



NOTE: Because the JUNOS Software 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.

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

Related Topics dynamic-db

prefix-list

Syntax	<pre>prefix-list <i>name</i> { <i>ip-addresses</i>; apply-path <i>path</i>; }</pre>
Hierarchy Level	[edit dynamic policy-options], [edit logical-systems <i>logical-system-name</i> policy-options], [edit policy-options]
Release Information	Statement introduced before JUNOS Release 7.4. Support for configuration in the dynamic database introduced in JUNOS Release 9.5.
Description	Define a list of IPv4 or IPv6 address prefixes for use in a routing policy statement or firewall filter statement.
Options	<p><i>name</i>—Name that identifies the list of IPv4 or IPv6 address prefixes.</p> <p><i>ip-addresses</i>—List of IPv4 or IPv6 address prefixes, one IP address per line in the configuration.</p> <p>The remaining statement is explained separately in this chapter.</p>
Usage Guidelines	See “Configuring Prefix Lists for Use in Routing Policy Match Conditions” on page 117 and “Configuring Routing Policies and Policy Objects in the Dynamic Database” on page 66.
Required Privilege Level	routing—To view this statement in the configuration. routing-control—To add this statement to the configuration.
Related Topics	dynamic-db

prefix-list-filter

Syntax	<code>prefix-list-filter <i>prefix-list-name</i> <i>match-type</i> <actions>;</code>
Hierarchy Level	[edit logical-systems <i>logical-system-name</i> policy-options], [edit policy-options]
Release Information	Statement introduced before JUNOS Release 7.4.
Description	Evaluate a list of prefixes within a prefix list using specified qualifiers.
Options	<p><i>prefix-list-name</i>—Name of the prefix list to evaluate.</p> <p><i>match-type</i>—Prefix length qualifiers.</p> <p><actions>—(Optional) Actions to take on match.</p>
Usage Guidelines	See “Configuring Prefix Lists for Use in Routing Policy Match Conditions” on page 117.
Required Privilege Level	<p>routing—To view this statement in the configuration.</p> <p>routing-control—To add this statement to the configuration.</p>

Part 3

Firewall Filters

- Introduction to Firewall Filters on page 173
- Firewall Filter Configuration on page 177
- Policer Overview on page 255
- Policer Configuration on page 257
- Summary of Firewall Filter and Policer Configuration Statements on page 287

Chapter 8

Introduction to Firewall Filters

This chapter describes the following topics:

- Firewall Filter Overview on page 173
- Firewall Filter Components on page 174
- Firewall Filter Types on page 175
- Supported Standards on page 176

Firewall Filter Overview

The basic purpose of a firewall filter is to enhance security through the use of packet filtering. The rules you define in a firewall filter are used to determine whether to accept, deny, or forward specific types of traffic. Firewall filters are stateless; they cannot statefully inspect traffic, that is keep track of the state of network connections.

The JUNOS Software firewall filters support a rich set of packet-matching criteria that you can use to match on specific traffic and perform specific actions, such as forwarding or dropping packets that match the criteria you specify. You can configure firewall filters to protect the local router or to protect another device that is either directly or indirectly connected to the local router. For example, you can use the filters to restrict the local packets that pass from the router's physical interfaces to the Routing Engine. Such filters are useful in protecting the IP services that run on the Routing Engine, such as Telnet, SSH, and BGP, from denial-of-service attacks.

You can also use firewall filters to perform multifield classification, counting, and policing. Multifield classification is used to perform specialized packet handling, including filter-based forwarding, or policy-based routing. Counting enables you to gather usage statistics. Policing is used to enforce bandwidth restrictions. Firewall filters that perform all these functions are *standard firewall filters*. The JUNOS Software also supports two additional specialized firewall filter types: *simple filters* and *service filters*.



NOTE: There is no limit to the number of filters and counters you can set, but there are some practical considerations. More counters require more terms, and a large number of terms can take a long time to process during a commit operation. However, filters with more than 4000 terms and counters have been implemented successfully.

Firewall Filter Components

A firewall filter consists of a protocol family and one or more terms that specify the filtering criteria and the action to take if a match occurs. After you define a firewall filter, you apply it to specific interfaces. Because the firewall filter process consists of two aspects—creating filters and then applying them—you can reuse the same filters on your router. Also, when you need to update the firewall filter itself, you have to make the change only in one place.

Protocol Family

When writing a firewall filter, you start by selecting the protocol family for which you want to specify filtering criteria. Firewall filters support the following protocol families:

- IPv4 (**inet**)
- IPv6 (**inet6**)
- MPLS (**mpls**)
- VPLS (**vpls**)
- Circuit cross-connects (**ccc**)
- (MX Series Ethernet Services routers only) Bridge (**bridge**)
- Protocol-independent (**any**)

Terms

Firewall filters require that you use *terms*. Each term can include both match criteria and actions.

The order in which you configure firewall filter terms is important. Terms are evaluated in the order in which they are configured. By default, new terms are always added to the end of the existing filter. You can use the **insert** command to reorder the terms of a firewall filter.

By default, each firewall filter ends with an implicit deny-all term. The final default action is to discard all packets. Packets that do not match any of the configured match conditions in a firewall filter are silently discarded.

If a packet arrives on an interface and a firewall filter is not configured for the incoming traffic on that interface, the packet is accepted by default.

Match Conditions

Match conditions are the fields or values that the packet must contain. You can define various match conditions, including the IP source address field, IP destination address field, TCP or User Datagram Protocol UDP source port field, IP protocol field, Internet Control Message Protocol (ICMP) packet type, IP options, TCP flags, incoming logical or physical interface, and outgoing logical or physical interface.

Actions

Within a single term, all the match conditions configured must match the packet before the configured action is taken on the packet. For a single match condition configured with multiple values, such as a range of values, only one of the values must match the packet before the match occurs and the configured action is taken on the packet.

Actions fall into the following categories:

- **Terminating**—A terminating action halts all evaluation of a firewall filter for a specific packet. The router performs the specified action, and no additional terms are examined.
- **Nonterminating**
 - **Actions**—Nonterminating actions are used to perform other functions on a packet, such as incrementing a counter, logging information about the packet header, sampling the packet data, or sending information to a remote host using the system log functionality.
 - **Next Term**—The action **next term** enables the router to perform configured actions on the packet and then evaluate the following term in the filter, rather than terminating the filter. If the **next term** action is included, the matching packet is then evaluated against the next term in the firewall filter; otherwise, the matching packet is not evaluated against subsequent terms in the firewall filter. For example, when you configure a term with the action modifier **count**, the term's action changes from an implicit **discard** to an implicit **accept**. The **next term** action forces the continued evaluation of the firewall filter.

Terminating and nonterminating actions that are configured within a single term are all taken on traffic that matches the conditions configured.

Attachment Points

After you define the firewall filter, you must apply it to an attachment point. These attachment points include logical interfaces, physical interfaces, routing interfaces and routing instances. You can apply a firewall filter as an *input* filter or an *output* filter, or both at the same time. Input filters take action on packets being received on the specified interface, whereas output filters take action on packets that are transmitted through the specified interface. You typically apply one filter with multiple terms to a single logical interface, to incoming traffic, outbound traffic, or both. However, there are times when you might want to chain multiple firewall filters (with single or multiple terms) together and apply them to an interface. You use an *input list* to apply multiple firewall filters to the incoming traffic on an interface. You use an *output list* to apply multiple firewall filters to the outbound traffic on an interface. You can include up to 16 filters in an input or an output list.

Firewall Filter Types

In addition to standard firewall filters, the JUNOS Software firewall filter implementation also supports two other firewall filter types: service filters and simple filters.

Service Filters

Service filters enable you to define filters associated with a defined set of services. Service filters are supported on services interfaces, which provide specific capabilities for manipulating traffic before it is delivered to its destination. You use service filters to refine the target of the set of services and also to process traffic. Only IPv4 and IPv6 traffic are supported on service filters. No other protocol families are supported.

Simple Filters

Simple filters are supported on Gigabit Ethernet intelligent queuing (IQ2) and Enhanced Queuing Dense Port Concentrator (EQ DPC) interfaces only. Unlike standard filters, simple filters support IPv4 traffic only and have a number of restrictions. For example, you cannot configure a terminating action for a simple filter. Simple filters always accept packets. Also, simple filters can be applied only as input filters. They are not supported on outbound traffic. Simple filters are recommended for metropolitan Ethernet applications.

Supported Standards

The JUNOS Software supports the following RFCs related to filtering:

- RFC 792, *Internet Control Message Protocol*
- RFC 2460, *Internet Protocol, Version 6 (IPv6)*
- RFC 2474, *Definition of the Differentiated Services (DS) Field*
- RFC 2475, *An Architecture for Differentiated Services*
- RFC 2597, *Assured Forwarding PHB Group*
- RFC 3246, *An Expedited Forwarding PHB (Per-Hop Behavior)*
- RFC 4291, *IP Version 6 Addressing Architecture*
- RFC 4443, *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*

Chapter 9

Firewall Filter Configuration

This chapter describes the following tasks for configuring firewall filters:

- Configuring Firewall Filters on page 178
- Configuring Standard Firewall Filters on page 179
- How Firewall Filters Are Evaluated on page 182
- Overview of Match Conditions in Firewall Filter Terms on page 183
- Configuring IPv4 Match Conditions on page 183
- Configuring IPv6 Match Conditions on page 187
- Configuring Protocol-Independent Match Conditions on page 191
- Configuring Layer 2 Circuit Cross-Connect Match Conditions on page 191
- Configuring MPLS Match Conditions on page 192
- Configuring VPLS Match Conditions on page 193
- Configuring Layer 2 Bridging Match Conditions for MX Series Ethernet Services Routers on page 197
- Overview of Protocol Match Conditions on page 200
- Example: Matching on Destination Port and Protocol Fields on page 200
- Overview of Class-Based Match Conditions on page 201
- How to Specify Firewall Filter Match Conditions on page 202
- Configuring Actions in Firewall Filter Terms on page 208
- Configuring Nested Firewall Filters on page 213
- Applying Firewall Filters to Interfaces on page 215
- Overview of Firewall Filter Lists on page 219
- Firewall Filter Examples on page 223
- Example: Blocking Telnet and SSH Access on page 223
- Example: Blocking TFTP Access on page 224
- Example: Accepting DHCP Packets with Specific Addresses on page 225
- Example: Defining a Policer for a Destination Class on page 225
- Example: Counting IP Option Packets on page 226
- Example: Accepting OSPF Packets from Certain Addresses on page 227
- Example: Matching Packets Based on Two Unrelated Criteria on page 227

- Example: Counting Both Accepted and Rejected Packets on page 228
- Example: Blocking TCP Connections to a Certain Port Except from BGP Peers on page 228
- Example: Accepting Packets with Specific IPv6 TCP Flags on page 229
- Example: Setting a Rate Limit for Incoming Layer 2 Control Packets on page 230
- Configuring Service Filters on page 231
- Configuring Simple Filters on page 232
- Configuring Firewall Filters for Logical Systems on page 234
- Configuring Accounting for Firewall Filters on page 247
- Configuring Filter-Based Forwarding on page 248
- Configuring Forwarding Table Filters on page 250
- Configuring System Logging of Firewall Filter Operations on page 252

Configuring Firewall Filters

This section shows the complete set of statements that can be configured at the [edit firewall] hierarchy level to create a firewall filter.

```
[edit firewall]
family family-name {
  filter filter-name {
    accounting-profile name;
    interface-specific;
    physical-interface-filter;
    term term-name {
      filter filter-name;
      from {
        match-conditions;
      }
      then {
        action;
        action-modifiers;
      }
    }
  }
}
service-filter filter-name {
  term term-name {
    from {
      match-conditions;
    }
    then {
      action;
      action-modifiers;
    }
  }
}
simple-filter filter-name {
  term term-name {
    from {
```

```

        match-conditions;
    }
    then {
        action;
        action-modifiers;
    }
}
}
}

```

To configure an IPv4 firewall filter, you can configure the filter at the [edit firewall] hierarchy level without including the **family inet** statement. The [edit firewall] and [edit firewall filter family inet] hierarchies are equivalent. The **family family-name** statement is required only to specify a protocol family other than IPv4.



NOTE: For stateless firewall filtering, you must allow the output tunnel traffic through the firewall filter applied to input traffic on the interface that is the next-hop interface towards the tunnel destination. The firewall filter affects only the packets exiting the router by way of the tunnel.

Configuring Standard Firewall Filters

When you configure a standard firewall filter, you must configure the following components:

- Protocol family for which you want to filter traffic.
- Filter name.
- At least one term, with a unique name for each term. A term is used to define match conditions that specify the fields or values that a packet must contain and actions to perform on traffic that matches the specified conditions.
- One or more match conditions for each term.
- Action for each term (recommended, because otherwise, packets are automatically accepted if they meet the configured match conditions).

To configure a firewall filter:

1. Include the **family** *family-name* statement at the [edit firewall] hierarchy level to specify the protocol family for which you want to filter traffic.

For *family-name*, specify one of the following:

- **inet**—IPv4
- **inet6**—IPv6
- **ccc**—Layer 2 circuit cross-connects
- **mpls**—MPLS
- **any**—Protocol-independent (Use this protocol family to apply a firewall filter to a physical interface.)
- **vpls**—VPLS
- **bridge**—(MX Series Ethernet Services Routers only) Layer 2 bridging

2. Include the **filter** *filter-name* statement at the [edit firewall family *family-name*] hierarchy level to specify a name for the firewall filter.

The filter name can contain letters, numbers, and hyphens (-) and be up to 64 characters long. To include spaces in the name, enclose the entire name in quotation marks (" ").

3. Include the **term** *term-name* statement at the [edit firewall family *family-name* filter *filter-name*] hierarchy level to configure a term.

Each firewall filter consists of one or more terms. For each term you specify one or more match conditions and one or more actions. The term name can contain letters, numbers, and hyphens (-) and can be up to 74 characters long. To include spaces in the name, enclose the entire name in quotation marks (" ").

You can specify multiple terms in a filter, effectively chaining together a series of match-action operations to apply to the packets on an interface.

4. Include the **from** *match-conditions* statement at the [edit firewall family *family-name* filter *filter-name* term *term-name*] hierarchy level to specify the fields or values that the packet must contain (match conditions).

For a match to occur, the packet must match all the conditions in the term. An individual match condition in a **from** statement can contain a list of values. For example, you can specify numeric range or multiple source and destination addresses. When a condition defines a list of values, a match occurs if any of the values matches the packet.



NOTE: The `from` statement is optional. If you omit it, the actions specified in the term's `then` statement are optional.

5. Include the `then actions` statement at the [edit firewall family *family-name* filter *filter-name* term *term-name* hierarchy level] to specify an action to perform on traffic that matches the conditions specified in the term.



BEST PRACTICE: We strongly recommend that you always explicitly configure an action in the `then` statement. If you do not, or if you omit the `then` statement entirely, packets that match the conditions in the `from` statement are automatically accepted.

You can specify the following filter actions:

- `accept`
- `count counter-name`
- `discard`
- `dscp code-point` (family inet only)
- `forwarding-class class-name`
- `ipsec-sa ipsec-sa` (family inet only)
- `load-balance group-name` (family inet only)
- `log` (family inet and inet6 only)
- `logical-system logical-system-name` (family inet and inet6 only)
- `loss-priority` (high | medium-high | medium-low | low)
- `next term`
- `next-hop-group group-name` (family inet only)
- `policer policer-name`
- `port-mirror` (family bridge, ccc, inet, inet6, and vpls only)
- `prefix-action action-name` (family inet only)
- `reject <message-type>` (family inet and inet6 only)
- `routing-instance routing-instance-name` (family inet and inet6 only)
- `sample` (family inet, inet6, and mpls only)
- `service-filter-hit` (service filters and family inet only)
- `syslog` (family inet and inet6 only)

- `three-color-policer` *policer-name*
- `topology` *topology-name* (family `inet` and `inet6` only)



NOTE: You can specify only one of the following actions in a single term: `accept`, `discard`, logical-system *logical-system-name*, `next term`, `reject`, routing-instance *routing-instance-name*, or topology *topology-name*. You can, however, specify one of these actions with one or more nonterminating actions in a single term. For example, within a term, you can specify `accept` with `count` and `syslog`.

- Related Topics**
- Overview of Match Conditions in Firewall Filter Terms on page 183
 - How to Specify Firewall Filter Match Conditions on page 202
 - Configuring Actions in Firewall Filter Terms on page 208

How Firewall Filters Are Evaluated

When a firewall filter consists of a single term, the filter is evaluated as follows:

- If the packet matches all the conditions, the action in the **then** statement is taken.
- If the packet matches all the conditions and if there is no action specified in the **then** statement, the default action **accept** is used.
- If the packet does not match all the conditions, it is discarded.

When a firewall filter consists of more than one term, the terms in the filter are evaluated sequentially:

1. The packet is evaluated against the conditions in the **from** statement in the first term.
2. If the packet matches the **from** statement, the action in the **then** statement is performed. Then:
 - If the **next term** action is not specified, the evaluation ends. Subsequent terms in the filter are not evaluated.
 - If the **next term** action is present, the evaluation continues to the next term.
3. If the packet does not match the **from** statement in the first term, it is evaluated against the conditions in the **from** statement in the second term.

This process continues until either the packet matches the **from** conditions in one of the subsequent terms or there are no more terms.

Both for filters with a single term and for filters with multiple terms, if a term does not contain a **from** statement, the action in the term's **then** statement is performed on all packets.

If a term does not contain a **then** statement or if you do not specify an action in the **then** statement, and if the packet matches the conditions in the term's **from** statement, the packet is accepted.

Each firewall filter has an implicit discard action at the end of the filter, which is equivalent to the following explicit filter term:

```
term implicit-rule {
  then discard;
}
```

Therefore, if a packet matches none of the terms in the filter, it is discarded.

Overview of Match Conditions in Firewall Filter Terms

In the **from** statement in a firewall filter term, you specify characteristics that the packet must have for the action in the subsequent **then** statement to be performed. The characteristics are referred to as *match conditions*. The packet must match all conditions in the **from** statement for the action to be performed, which also means that their order in the **from** statement is not important.

Each protocol family supports a different set of match conditions, and some match conditions are supported only on certain routers. For example, a number of match conditions for VPLS traffic are supported only on the MX Series Ethernet Services Routers.

Configuring IPv4 Match Conditions

Table 24 on page 183 describes the firewall filter match conditions that are supported for IPv4 traffic.

To configure firewall filter match conditions for IPv4 traffic:

- Include the *match-conditions* statement at the [edit firewall family *family-name* filter *filter-name* term *term-name* from]

Table 24: IPv4 Firewall Filter Match Conditions

Match Condition	Description
<i>keyword-except</i>	Negate a match. For example, <i>destination-port-except number</i> .
<i>ah-spi spi-value</i>	IPsec authentication header (AH) security parameter index (SPI) value. Match on this specific SPI value.
<i>ah-spi-except spi-value</i>	IPsec AH SPI value. Do not match on this specific SPI value.
<i>destination-address address</i>	Destination prefix.
<i>destination-class class-name</i>	One or more destination classes

Table 24: IPv4 Firewall Filter Match Conditions (continued)

Match Condition	Description
<code>destination-mac-address address</code>	Destination media access control (MAC) address of a VPLS packet.
<code>destination-port number</code>	<p>TCP or User Datagram Protocol (UDP) destination port field. You cannot specify both the port and destination-port match conditions in the same term.</p> <p>Normally, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more information, see “Overview of Protocol Match Conditions” on page 200.</p> <p>In place of the numeric value, you can specify one of the following text synonyms (the port numbers are also listed): afs (1483), bgp (179), biff (512), bootpc (68), bootps (67), cmd (514), cvspserver (2401), dhcp (67), domain (53), eklogin (2105), ekshell (2106), exec (512), finger (79), ftp (21), ftp-data (20), http (80), https (443), ident (113), imap (143), kerberos-sec (88), klogin (543), kpasswd (761), krb-prop (754), krbupdate (760), kshell (544), ldap (389), login (513), mobileip-agent (434), mobilip-mn (435), msdp (639), netbios-dgm (138), netbios-ns (137), netbios-ssn (139), nfsd (2049), nnpt (119), ntalk (518), ntp (123), pop3 (110), pptp (1723), printer (515), radacct (1813), radius (1812), rip (520), rinit (2108), smtp (25), snmp (161), snmptrap (162), snpp (444), socks (1080), ssh (22), sunrpc (111), syslog (514), tacacs-ds (65), talk (517), telnet (23), tftp (69), timed (525), who (513), xmcp (177).</p>
<code>destination-prefix-list name</code>	Destination prefixes in the specified list name. Specify the name of a prefix list defined at the [edit policy-options prefix-list <i>prefix-list-name</i>] hierarchy level.
<code>dscp number</code>	<p>Differentiated Services code point (DSCP). The DiffServ protocol uses the type-of-service (ToS) byte in the IP header. The most significant 6 bits of this byte form the DSCP. For more information, see the <i>JUNOS Class of Service Configuration Guide</i>.</p> <p>You can specify DSCP in hexadecimal, binary, or decimal form.</p> <p>In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed):</p> <ul style="list-style-type: none"> ■ RFC 3246, <i>An Expedited Forwarding PHB (Per-Hop Behavior)</i>, defines one code point: ef (46). ■ RFC 2597, <i>Assured Forwarding PHB Group</i>, defines 4 classes, with 3 drop precedences in each class, for a total of 12 code points: <ul style="list-style-type: none"> ■ af11 (10), af12 (12), af13 (14) ■ af21 (18), af22 (20), af23 (22) ■ af31 (26), af32 (28), af33 (30) ■ af41 (34), af42 (36), af43 (38)
<code>ether-type value</code>	Ethernet type field of a VPLS packet.
<code>ether-type-except value</code>	Do not match on the Ethernet type field of a VPLS packet.
<code>esp-spi spi-value</code>	IPsec encapsulating security payload (ESP) SPI value. Match on this specific SPI value. You can specify the ESP SPI value in hexadecimal, binary, or decimal form.
<code>esp-spi-except spi-value</code>	IPsec ESP SPI value. Do not match on this specific SPI value.
<code>first-fragment</code>	First fragment of a fragmented packet. This condition does not match unfragmented packets.
<code>forwarding-class class</code>	Forwarding class. Specify assured-forwarding , best-effort , expedited-forwarding , or network-control .

Table 24: IPv4 Firewall Filter Match Conditions (*continued*)

Match Condition	Description
<code>forwarding-class-except class</code>	Do not match on the forwarding class. Specify <code>assured-forwarding</code> , <code>best-effort</code> , <code>expedited-forwarding</code> , or <code>network-control</code> .
<code>fragment-flags number</code>	IP fragmentation flags. In place of the numeric field value, you can specify one of the following keywords (the field values are also listed): <code>dont-fragment</code> (0x4000), <code>more-fragments</code> (0x2000), or <code>reserved</code> (0x8000).
<code>fragment-offset number</code>	Fragment offset field.
<code>icmp-code number</code>	<p>ICMP code field. This value or keyword provides more specific information than <code>icmp-type</code>. Because the value's meaning depends upon the associated <code>icmp-type</code>, you must specify <code>icmp-type</code> along with <code>icmp-code</code>. For more information, see “Overview of Protocol Match Conditions” on page 200.</p> <p>In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed). The keywords are grouped by the ICMP type with which they are associated:</p> <ul style="list-style-type: none"> ■ <code>parameter-problem</code>: <code>ip-header-bad</code> (0), <code>required-option-missing</code> (1) ■ <code>redirect</code>: <code>redirect-for-host</code> (1), <code>redirect-for-network</code> (0), <code>redirect-for-tos-and-host</code> (3), <code>redirect-for-tos-and-net</code> (2) ■ <code>time-exceeded</code>: <code>ttl-eq-zero-during-reassembly</code> (1), <code>ttl-eq-zero-during-transit</code> (0) ■ <code>unreachable</code>: <code>communication-prohibited-by-filtering</code> (13), <code>destination-host-prohibited</code> (10), <code>destination-host-unknown</code> (7), <code>destination-network-prohibited</code> (9), <code>destination-network-unknown</code> (6), <code>fragmentation-needed</code> (4), <code>host-precedence-violation</code> (14), <code>host-unreachable</code> (1), <code>host-unreachable-for-TOS</code> (12), <code>network-unreachable</code> (0), <code>network-unreachable-for-TOS</code> (11), <code>port-unreachable</code> (3), <code>precedence-cutoff-in-effect</code> (15), <code>protocol-unreachable</code> (2), <code>source-host-isolated</code> (8), <code>source-route-failed</code> (5)
<code>icmp-type number</code>	<p>ICMP packet type field. Normally, you specify this match in conjunction with the <code>protocol</code> match statement to determine which protocol is being used on the port. For more information, see “Overview of Protocol Match Conditions” on page 200.</p> <p>In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): <code>echo-reply</code> (0), <code>echo-request</code> (8), <code>info-reply</code> (16), <code>info-request</code> (15), <code>mask-request</code> (17), <code>mask-reply</code> (18), <code>parameter-problem</code> (12), <code>redirect</code> (5), <code>router-advertisement</code> (9), <code>router-solicit</code> (10), <code>source-quench</code> (4), <code>time-exceeded</code> (11), <code>timestamp</code> (13), <code>timestamp-reply</code> (14), or <code>unreachable</code> (3).</p>
<code>interface interface-name</code>	Interface on which the packet was received. You can configure a match condition that matches packets based on the interface on which they were received.
<code>interface-group group-number</code>	Interface group on which the packet was received. An interface group is a set of one or more logical interfaces. For <code>group-number</code> , specify a value from 0 through 255. For information about configuring interface groups, see “Applying Firewall Filters to Interfaces” on page 215.
<code>interface-set interface-set-name</code>	(MX Series routers and routers with Enhanced IQ2 [IQ2E] PICs only) Interface set on which the packet was received. An interface set is a set of logical interfaces used to configure hierarchical class-of-service schedulers. For information about configuring an interface set, see the <i>JUNOS Class of Service Configuration Guide</i> and the <i>JUNOS Network Interfaces Configuration Guide</i> .
<code>ip-options number</code>	IP options. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): <code>any</code> , <code>loose-source-route</code> (131), <code>route-record</code> (7), <code>router-alert</code> (148), <code>security</code> (130), <code>stream-id</code> (136), <code>strict-source-route</code> (137), or <code>timestamp</code> (68).
<code>is-fragment</code>	This condition matches if the packet is a trailing fragment; it does not match the first fragment of a fragmented packet. To match both first and trailing fragments, you can use two terms.

Table 24: IPv4 Firewall Filter Match Conditions (*continued*)

Match Condition	Description
<i>loss-priority level</i>	<p>Packet loss priority (PLP) level. Specify a single level or multiple levels: low, medium-low, medium-high, or high.</p> <p>Supported on MX Series routers; M120 and M320 routers; and M7i and M10i routers with the Enhanced CFEB (CFEB-E).</p> <p>On M320 routers, you must enable the tricolor statement at the [edit class-of-service] hierarchy level to commit a PLP configuration with any of the four levels specified. If the tricolor statement is not referenced, you can only configure the high and low levels. This applies to all protocol families.</p> <p>For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see the <i>JUNOS Class of Service Configuration Guide</i>.</p>
<i>loss-priority-except level</i>	<p>Do not match on the packet loss priority level. Specify a single level or multiple levels: low, medium-low, medium-high, or high.</p> <p>For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see the <i>JUNOS Class of Service Configuration Guide</i>.</p>
<i>packet-length bytes</i>	<p>Length of the received packet, in bytes. The length refers only to the IP packet, including the packet header, and does not include any Layer 2 encapsulation overhead.</p>
<i>port number</i>	<p>TCP or UDP source or destination port field. You cannot specify both the port match and either the destination-port or source-port match conditions in the same term.</p> <p>Normally, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more information, see “Overview of Protocol Match Conditions” on page 200.</p> <p>In place of the numeric value, you can specify one of the text synonyms listed under destination-port.</p>
<i>precedence ip-precedence-field</i>	<p>IP precedence field. In place of the numeric field value, you can specify one of the following text synonyms (the field values are also listed): critical-ecp (0xa0), flash (0x60), flash-override (0x80), immediate (0x40), internet-control (0xc0), net-control (0xe0), priority (0x20), or routine (0x00). You can specify precedence in hexadecimal, binary, or decimal form.</p>
<i>prefix-list name</i>	<p>Destination or source prefixes in the specified list name. Specify the name of a prefix list defined at the [edit policy-options prefix-list prefix-list-name] hierarchy level.</p>
<i>protocol number</i>	<p>IP protocol field. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): ah (51), egp (8), esp (50), gre (47), icmp (1), igmp (2), ipip (4), ipv6 (41), ospf (89), pim (103), rsvp (46), tcp (6), or udp (17).</p>
<i>source-class class-name</i>	<p>One or more source-class names.</p>
<i>source-port number</i>	<p>TCP or UDP source port field. You cannot specify the port and source-port match conditions in the same term.</p> <p>Normally, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more information, see “Overview of Protocol Match Conditions” on page 200.</p> <p>In place of the numeric field, you can specify one of the text synonyms listed under destination-port.</p>

Table 24: IPv4 Firewall Filter Match Conditions (*continued*)

Match Condition	Description
source-prefix-list <i>name</i>	Source prefixes in the specified list name. Specify the name of a prefix list defined at the [edit policy-options prefix-list <i>prefix-list-name</i>] hierarchy level.
tcp-established	TCP packets other than the first packet of a connection. This is a synonym for "(ack rst)". This condition does not implicitly check that the protocol is TCP. To check this, specify the protocol tcp match condition.
tcp-flags <i>number</i>	TCP flags. Normally, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more details, see "Overview of Protocol Match Conditions" on page 200. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): ack (0x10), fin (0x01), push (0x08), rst (0x04), syn (0x02), or urgent (0x20).
tcp-initial	First TCP packet of a connection. This is a synonym for "(syn & !ack)". This condition does not implicitly check that the protocol is TCP. To check this, specify the protocol tcp match condition.
ttl <i>number</i>	IPv4 time-to-live number. Specify a TTL value or a range of TTL values. For <i>number</i> , you can specify one or more values from 0 through 255. This match condition is supported only on M120, M320, MX Series, and T Series routers.
ttl-except <i>number</i>	Do not match on the IPv4 TTL number. Specify a TTL value or a range of values. For <i>number</i> , you can specify one or more values from 0 through 255. This match condition is supported only on M120, M320, MX Series, and T Series routers.
vlan-ether-type <i>value</i>	Virtual local area network (VLAN) Ethernet type field of a VPLS packet.
vlan-ether-type-except <i>value</i>	Do not match on the VLAN Ethernet type field of a VPLS packet.

- Related Topics**
- How to Specify Firewall Filter Match Conditions on page 202
 - Overview of Protocol Match Conditions on page 200
 - Overview of Class-Based Match Conditions on page 201

Configuring IPv6 Match Conditions

Table 25 on page 188 describes the firewall filter match conditions supported for IPv6 traffic.

To configure firewall filter match conditions for IPv6 traffic:

- Include the *match-conditions* statement at the [edit firewall family inet6 filter *filter-name* term *term-name* from] hierarchy level.

Table 25: IPv6 Firewall Filter Match Conditions

Match Condition	Description
<i>address address</i>	128-bit address that supports the standard syntax for IPv6 addresses. For more information, see the <i>JUNOS Routing Protocols Configuration Guide</i> .
<i>destination-address address</i>	128-bit address that is the final destination node address for the packet. The filter description syntax supports the text representations for IPv6 addresses as described in RFC 2373, <i>IP Version 6 Addressing Architecture</i> . For more information about IPv6 address syntax, see the <i>JUNOS Routing Protocols Configuration Guide</i> .
<i>destination-port number</i>	<p>TCP or UDP destination port field. You cannot specify both the <i>port</i> and <i>destination-port</i> match conditions in the same term.</p> <p>Normally, you specify this match in conjunction with the <i>next-header</i> match statement to determine which protocol is being used on the port. For more information, see “Overview of Protocol Match Conditions” on page 200.</p> <p>In place of the numeric value, you can specify one of the following text synonyms (the port numbers are also listed): <i>afs</i> (1483), <i>bgp</i> (179), <i>biff</i> (512), <i>bootpc</i> (68), <i>bootps</i> (67), <i>cmd</i> (514), <i>cvspserver</i> (2401), <i>dhcp</i> (67), <i>domain</i> (53), <i>eklogin</i> (2105), <i>ekshell</i> (2106), <i>exec</i> (512), <i>finger</i> (79), <i>ftp</i> (21), <i>ftp-data</i> (20), <i>http</i> (80), <i>https</i> (443), <i>ident</i> (113), <i>imap</i> (143), <i>kerberos-sec</i> (88), <i>klogin</i> (543), <i>kpasswd</i> (761), <i>krb-prop</i> (754), <i>krbupdate</i> (760), <i>kshell</i> (544), <i>ldap</i> (389), <i>login</i> (513), <i>mobileip-agent</i> (434), <i>mobileip-mn</i> (435), <i>msdp</i> (639), <i>netbios-dgm</i> (138), <i>netbios-ns</i> (137), <i>netbios-ssn</i> (139), <i>nfsd</i> (2049), <i>nntp</i> (119), <i>ntalk</i> (518), <i>ntp</i> (123), <i>pop3</i> (110), <i>pptp</i> (1723), <i>printer</i> (515), <i>radacct</i> (1813), <i>radius</i> (1812), <i>rip</i> (520), <i>rkinit</i> (2108), <i>smtp</i> (25), <i>snmp</i> (161), <i>snmptrap</i> (162), <i>snpp</i> (444), <i>socks</i> (1080), <i>ssh</i> (22), <i>sunrpc</i> (111), <i>syslog</i> (514), <i>tacacs-ds</i> (65), <i>talk</i> (517), <i>telnet</i> (23), <i>tftp</i> (69), <i>timed</i> (525), <i>who</i> (513), <i>xmcp</i> (177), <i>zephyr-clt</i> (2103), or <i>zephyr-hm</i> (2104).</p>
<i>destination-prefix-list name</i>	Destination prefixes in the specified list name. Specify the name of a prefix list defined at the [edit policy-options prefix-list <i>prefix-list-name</i>] hierarchy level.
<i>forwarding-class class</i>	Forwarding class. Specify <i>assured-forwarding</i> , <i>best-effort</i> , <i>expedited-forwarding</i> , or <i>network-control</i> .
<i>icmp-code number</i>	<p>ICMP code field. This value or keyword provides more specific information than <i>icmp-type</i>. Because the value’s meaning depends upon the associated <i>icmp-type</i>, you must specify <i>icmp-type</i> along with <i>icmp-code</i>. For more information, see “Overview of Protocol Match Conditions” on page 200.</p> <p>In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed). The keywords are grouped by the ICMP type with which they are associated:</p> <ul style="list-style-type: none"> ■ parameter-problem: <i>ip6-header-bad</i> (0), <i>unrecognized-next-header</i> (1), <i>unrecognized-option</i> (2) ■ time-exceeded: <i>ttl-eq-zero-during-reassembly</i> (1), <i>ttl-eq-zero-during-transit</i> (0) ■ destination-unreachable: <i>no-route-to-destination</i> (0), <i>administratively-prohibited</i> (1), <i>address-unreachable</i> (3), <i>port-unreachable</i> (4)

Table 25: IPv6 Firewall Filter Match Conditions (continued)

Match Condition	Description
<code>icmp-type number</code>	<p>ICMP packet type field. Normally, you specify this match in conjunction with the <code>protocol</code> match statement to determine which protocol is being used on the port. For more information, see “Overview of Protocol Match Conditions” on page 200.</p> <p>In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): <code>echo-reply</code> (129), <code>echo-request</code> (128), <code>membership-query</code> (130), <code>membership-report</code> (131), <code>membership-termination</code> (132), <code>neighbor-advertisement</code> (136), <code>neighbor-solicit</code> (135), <code>node-information-reply</code> (140), <code>node-information-request</code> (139), <code>packet-too-big</code> (2), <code>parameter-problem</code> (4), <code>redirect</code> (137), <code>router-advertisement</code> (134), <code>router-renumbering</code> (138), <code>router-solicit</code> (133), <code>time-exceeded</code> (3), or <code>destination-unreachable</code> (1).</p>
<code>interface interface-name</code>	Interface on which the packet was received. You can configure a match condition that matches packets based on the interface on which they were received.
<code>interface-group group-number</code>	Interface group on which the packet was received. An interface group is a set of one or more logical interfaces. For information about configuring interface groups, see “Applying Firewall Filters to Interfaces” on page 215.
<code>interface-set interface-set-name</code>	(MX Series routers and routers with Enhanced IQ2 [IQ2E] PICs only) Interface set on which the packet was received. An interface set is a set of logical interfaces used to configure hierarchical class-of-service schedulers. For information about configuring an interface set, see the <i>JUNOS Class of Service Configuration Guide</i> and the <i>JUNOS Network Interfaces Configuration Guide</i> .
<code>loss-priority level</code>	<p>Packet loss priority (PLP) level. Specify a single level or multiple levels: <code>low</code>, <code>medium-low</code>, <code>medium-high</code>, or <code>high</code>.</p> <p>Supported on MX Series routers; M120 and M320 routers; and M7i and M10i routers with the Enhanced CFEB (CFEB-E).</p> <p>On M320 routers, you must enable the <code>tricolor</code> statement at the <code>[edit class-of-service]</code> hierarchy level to commit a PLP configuration with any of the four levels specified. If the <code>tricolor</code> statement is not referenced, you can only configure the <code>high</code> and <code>low</code> levels. This applies to all protocol families.</p> <p>For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see the <i>JUNOS Class of Service Configuration Guide</i>.</p>
<code>loss-priority-except level</code>	<p>Do not match on the packet loss priority level. Specify a single level or multiple levels: <code>low</code>, <code>medium-low</code>, <code>medium-high</code>, or <code>high</code>.</p> <p>For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see the <i>JUNOS Class of Service Configuration Guide</i>.</p>
<code>next-header bytes</code>	8-bit IP protocol field that identifies the type of header immediately following the IPv6 header. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): <code>ah</code> (51), <code>dstops</code> (60), <code>egp</code> (8), <code>esp</code> (50), <code>fragment</code> (44), <code>gre</code> (47), <code>hop-by-hop</code> (0), <code>icmp</code> (1), <code>icmpv6</code> (1), <code>igmp</code> (2), <code>ipip</code> (4), <code>ipv6</code> (41), <code>no-next-header</code> (59), <code>ospf</code> (89), <code>pim</code> (103), <code>routing</code> (43), <code>rsvp</code> (46), <code>sctp</code> (132), <code>tcp</code> (6), <code>udp</code> (17), or <code>vrrp</code> (112).
<code>packet-length bytes</code>	Length of the received packet, in bytes. The length refers only to the IP packet, including the packet header, and does not include any Layer 2 encapsulation overhead.

Table 25: IPv6 Firewall Filter Match Conditions (*continued*)

Match Condition	Description
<i>port number</i>	<p>TCP or UDP source or destination port field. You cannot specify both the port match and either the destination-port or source-port match conditions in the same term.</p> <p>Typically, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more information, see “Overview of Protocol Match Conditions” on page 200.</p> <p>In place of the numeric value, you can specify one of the text synonyms listed under destination-port.</p>
<i>prefix-list name</i>	Source or destination prefixes in the specified list name. Specify the name of a list defined at the [edit routing-options prefix-list <i>prefix-list-name</i>] hierarchy level.
<i>source-address address</i>	Address of the source node sending the packet; 128 bits in length. The filter description syntax supports the text representations for IPv6 addresses as described in RFC 2373. For more information about IPv6 address syntax, see the <i>JUNOS Routing Protocols Configuration Guide</i> .
<i>source-port number</i>	<p>TCP or UDP source port field. You cannot specify the port and source-port match conditions in the same term.</p> <p>Normally, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more information, see “Overview of Protocol Match Conditions” on page 200.</p> <p>In place of the numeric field, you can specify one of the text synonyms listed under destination-port.</p>
<i>source-prefix-list name</i>	Source prefixes in the specified prefix list. Specify a prefix list name defined at the [edit policy-options prefix-list <i>prefix-list-name</i>] hierarchy level.
<i>tcp-established</i>	<p>TCP packets other than the first packet of a connection. This is a synonym for "(ack rst)".</p> <p>This condition does not implicitly check that the protocol is TCP. To check this, specify the protocol tcp match condition.</p>
<i>tcp-flags flags</i>	<p>One or more of the following TCP flags:</p> <ul style="list-style-type: none"> ■ bit-name: fin, syn, rst, push, ack, urgent You can string together multiple flags using logical operators. ■ numerical value: 0x01 through 0x20 ■ text synonym: tcp-established, tcp-initial <p>Configuring the tcp-flags match condition requires that you configure the next-header tcp match condition.</p>
<i>tcp-initial</i>	Initial packet of a TCP connection. Configuring the tcp-initial match condition also requires that you to configure the next-header match condition.
<i>traffic-class number</i>	<p>8-bit field that specifies the class-of-service (CoS) priority of the packet. The traffic-class field is used to specify a DiffServ code point (DSCP) value. The numerical value cannot be greater than 0x3f.</p> <p>This field was previously used as the ToS field in IPv4. However, the semantics of this field (for example, DSCP) are identical to those of IPv4.</p>

Related Topics ■ How to Specify Firewall Filter Match Conditions on page 202

- Overview of Protocol Match Conditions on page 200
- Overview of Class-Based Match Conditions on page 201

Configuring Protocol-Independent Match Conditions

Table 26 on page 191 describes the firewall filter match conditions for protocol-independent traffic.

To configure firewall filter match conditions for protocol-independent traffic:

- Include the *match-conditions* statement at the [edit firewall family any filter *filter-name* term *term-name* from] hierarchy level.

Table 26: Protocol-Independent Firewall Filter Match Conditions

Match Condition	Description
forwarding-class <i>class</i>	Forwarding class. Specify assured-forwarding, best-effort, expedited-forwarding, or network-control.
forwarding-class-except <i>class</i>	Do not match on the forwarding class. Specify assured-forwarding, best-effort, expedited-forwarding, or network-control.
interface <i>interface-name</i>	Interface on which the packet was received. You can configure a match condition that matches packets based on the interface on which they were received.
interface-set <i>interface-set-name</i>	(MX Series routers and routers with Enhanced IQ2 [IQ2E] PICs only) Interface set on which the packet was received. An interface set is a set of logical interfaces used to configure hierarchical class of service schedulers. For information about configuring an interface set, see the <i>JUNOS Class of Service Configuration Guide</i> and the <i>JUNOS Network Interfaces Configuration Guide</i> .
packet-length <i>bytes</i>	Length of the received packet, in bytes. The length refers only to the IP packet, including the packet header, and does not include any Layer 2 encapsulation overhead.
packet-length-except <i>bytes</i>	Do not match on the received packet length, in bytes.

Configuring Layer 2 Circuit Cross-Connect Match Conditions

Table 27 on page 191 describes the firewall filter match conditions for Layer 2 circuit cross-connect (CCC) traffic.

To configure firewall filter match conditions for Layer 2 CCC traffic:

- Include the *match-conditions* statement at the [edit firewall family ccc filter *filter-name* term *term-name* from] hierarchy level.

Table 27: Layer 2 Circuit Cross-Connect Firewall Filter Match Conditions

Match Condition	Description
forwarding-class <i>class</i>	Forwarding class. Specify assured-forwarding, best-effort, expedited-forwarding, or network-control.

Table 27: Layer 2 Circuit Cross-Connect Firewall Filter Match Conditions (*continued*)

Match Condition	Description
<code>forwarding-class-except class</code>	Do not match on the forwarding class. Specify <code>assured-forwarding</code> , <code>best-effort</code> , <code>expedited-forwarding</code> , or <code>network-control</code> .
<code>interface-group group-number</code>	Interface group on which the packet was received. An interface group is a set of one or more logical interfaces. For <i>group-number</i> , specify a value from 0 through 255. For information about configuration interface groups, see “Applying Firewall Filters to Interfaces” on page 215.
<code>interface-group-except number</code>	Do not match on the interface group in which the packet was received. For <i>group-number</i> , specify a value from 0 through 255.
<code>loss-priority level</code>	<p>Packet loss priority (PLP) level. Specify a single level or multiple levels: <code>low</code>, <code>medium-low</code>, <code>medium-high</code>, or <code>high</code>.</p> <p>Supported on MX Series routers; M120 and M320 routers; and M7i and M10i routers with the Enhanced CFEB (CFEB-E).</p> <p>On M320 routers, you must enable the <code>tricolor</code> statement at the [edit class-of-service] hierarchy level to commit a PLP configuration with any of the four levels specified. If the <code>tricolor</code> statement is not referenced, you can only configure the <code>high</code> and <code>low</code> levels. This applies to all protocol families.</p> <p>For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see the <i>JUNOS Class of Service Configuration Guide</i>.</p>
<code>loss-priority-except level</code>	<p>Do not match on the packet loss priority level. Specify a single level or multiple levels: <code>low</code>, <code>medium-low</code>, <code>medium-high</code>, or <code>high</code>.</p> <p>For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see the <i>JUNOS Class of Service Configuration Guide</i>.</p>

Configuring MPLS Match Conditions

Table 28 on page 192 describes the firewall filter match conditions supported for MPLS traffic.

To configure firewall filter match conditions for MPLS traffic:

- Include the `match-conditions` statement at the [edit firewall family mpls filter *filter-name* term *term-name* from] hierarchy level.

Table 28: MPLS Firewall Filter Match Conditions

Match Condition	Description
<code>exp number</code>	Experimental (EXP) bit number or range of bit numbers in the MPLS header. For <i>number</i> , you can specify one or more values from 0 through 7 in decimal, binary, or hexadecimal format.
<code>exp-except number</code>	Do not match on the EXP bit number or range of bit numbers in the MPLS header. For <i>number</i> , you can specify one or more values from 0 through 7.
<code>forwarding-classclass</code>	Forwarding class. Specify <code>assured-forwarding</code> , <code>best-effort</code> , <code>expedited-forwarding</code> , or <code>network-control</code> .

Table 28: MPLS Firewall Filter Match Conditions (continued)

Match Condition	Description
forwarding-class-except class	Do not match on the forwarding class. Specify assured-forwarding , best-effort , expedited-forwarding , or network-control .
interface interface-name	Interface on which the packet was received. You can configure a match condition that matches packets based on the interface on which they were received.
interface-set interface-set-name	(MX Series routers and routers with Enhanced IQ2 [IQ2E] PICs only) Interface set on which the packet was received. An interface set is a set of logical interfaces used to configure hierarchical class-of-service schedulers. For information about configuring an interface set, see the <i>JUNOS Class of Service Configuration Guide</i> and the <i>JUNOS Network Interfaces Configuration Guide</i> .

Configuring VPLS Match Conditions

Table 29 on page 193 describes the firewall filter match conditions supported for VPLS.

Not all match conditions for VPLS traffic are supported on all routing platforms. A number of match conditions for VPLS traffic are supported only on MX Series Ethernet Services Routers, as noted in the Table 29 on page 193.

To configure firewall filter match conditions for VPLS traffic:

- Include the *match-conditions* statement at the [edit firewall family vpls filter *filter-name* term *term-name* from] hierarchy level.

For more information about how to configure Layer 2 services on the MX Series routers, see the *JUNOS Network Interfaces Configuration Guide*, the *JUNOS MX Series Ethernet Services Routers Layer 2 Configuration Guide*, and the *JUNOS MX Series Ethernet Services Routers Solutions Guide*.

Table 29: VPLS Firewall Filter Match Conditions

Match Condition	Description
destination mac-address address	Destination media access control (MAC) address of a VPLS packet.
destination-port number	(MX Series routers only) TCP or UDP destination port field. You cannot specify both the port and destination-port match conditions in the same term.
destination-port-except number	(MX Series routers only) Do not match on the TCP or UDP destination port field. You cannot specify both the port and destination-port match conditions in the same term.

Table 29: VPLS Firewall Filter Match Conditions (*continued*)

Match Condition	Description
<code>dscp number</code>	<p>(MX Series routers only) Differentiated Services code point (DSCP). The DiffServ protocol uses the type-of-service (ToS) byte in the IP header. The most significant 6 bits of this byte form the DSCP. For more information, see the <i>JUNOS Class of Service Configuration Guide</i>.</p> <p>You can specify DSCP in either hexadecimal, binary, or decimal form.</p> <p>In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed):</p> <ul style="list-style-type: none"> ■ RFC 3246, <i>An Expedited Forwarding PHB (Per-Hop Behavior)</i>, defines one code point: ef (46). ■ RFC 2597, <i>Assured Forwarding PHB Group</i>, defines 4 classes, with 3 drop precedences in each class, for a total of 12 code points: <p>af11 (10), af12 (12), af13 (14),</p> <p>af21 (18), af22 (20), af23 (22),</p> <p>af31 (26), af32 (28), af33 (30),</p> <p>af41 (34), af42 (36), af43 (38)</p>
<code>dscp-except number</code>	(MX Series routers only) Do not match on the DSCP.
<code>ether-type number</code>	Ethernet type field of a VPLS packet.
<code>ether-type-except number</code>	Do not match on the Ethernet type field of a VPLS packet.
<code>forwarding-class class</code>	Forwarding class. Specify assured-forwarding , best-effort , expedited-forwarding , or network-control .
<code>forwarding-class-except class</code>	Do not match on the forwarding class. Specify assured-forwarding , best-effort , expedited-forwarding , or network-control .
<code>icmp-code number</code>	<p>(MX Series routers only) ICMP code field. This value or keyword provides more specific information than icmp-type. Because the value's meaning depends upon the associated icmp-type, you must specify icmp-type along with icmp-code. For more information, see "Overview of Protocol Match Conditions" on page 200.</p> <p>In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed). The keywords are grouped by the ICMP type with which they are associated:</p> <ul style="list-style-type: none"> ■ parameter-problem: ip-header-bad (0), required-option-missing (1) ■ redirect: redirect-for-host (1), redirect-for-network (0), redirect-for-tos-and-host (3), redirect-for-tos-and-net (2) ■ time-exceeded: ttl-eq-zero-during-reassembly (1), ttl-eq-zero-during-transit (0) ■ unreachable: communication-prohibited-by-filtering (13), destination-host-prohibited (10), destination-host-unknown (7), destination-network-prohibited (9), destination-network-unknown (6), fragmentation-needed (4), host-precedence-violation (14), host-unreachable (1), host-unreachable-for-TOS (12), network-unreachable (0), network-unreachable-for-TOS (11), port-unreachable (3), precedence-cutoff-in-effect (15), protocol-unreachable (2), source-host-isolated (8), source-route-failed (5)
<code>icmp-code-except number</code>	(MX Series routers only) Do not match on the ICMP code field.

Table 29: VPLS Firewall Filter Match Conditions (*continued*)

Match Condition	Description
<code>icmp-type number</code>	<p>(MX Series routers only) ICMP packet type field. Normally, you specify this match in conjunction with the <code>protocol</code> match statement to determine which protocol is being used on the port. For more information, see “Overview of Protocol Match Conditions” on page 200.</p> <p>In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): <code>echo-reply</code> (0), <code>echo-request</code> (8), <code>info-reply</code> (16), <code>info-request</code> (15), <code>mask-request</code> (17), <code>mask-reply</code> (18), <code>parameter-problem</code> (12), <code>redirect</code> (5), <code>router-advertisement</code> (9), <code>router-solicit</code> (10), <code>source-quench</code> (4), <code>time-exceeded</code> (11), <code>timestamp</code> (13), <code>timestamp-reply</code> (14), or <code>unreachable</code> (3).</p>
<code>icmp-type-except number</code>	(MX Series routers only) Do not match on the ICMP packet type field.
<code>interface-group group-name</code>	Interface group on which the packet was received. An interface group is a set of one or more logical interfaces. For information about configuring interface groups, see “Applying Firewall Filters to Interfaces” on page 215.
<code>interface-group-except group-name</code>	Do not match on the interface group.
<code>ip-address address</code>	(MX Series routers only) 32-bit address that supports the standard syntax for IPv4 addresses.
<code>ip-destination-address address</code>	(MX Series routers only) 32-bit address that is the final destination node address for the packet.
<code>ip-precedence ip-precedence-field</code>	(MX Series routers only) IP precedence field. In place of the numeric field value, you can specify one of the following text synonyms (the field values are also listed): <code>critical-ecp</code> (0xa0), <code>flash</code> (0x60), <code>flash-override</code> (0x80), <code>immediate</code> (0x40), <code>internet-control</code> (0xc0), <code>net-control</code> (0xe0), <code>priority</code> (0x20), or <code>routine</code> (0x00).
<code>ip-precedence-except ip-precedence-field</code>	(MX Series routers only) Do not match on the IP precedence field.
<code>ip-protocol number</code>	(MX Series routers only) IP protocol field.
<code>ip-protocol-except number</code>	(MX Series routers only) Do not match on the IP protocol field.
<code>ip-source-address address</code>	(MX Series routers only) IP address of the source node sending the packet.
<code>learn-vlan-1p-priority number</code>	(MX Series routers only) IEEE 802.1p learned VLAN priority field. Specify a single value or multiple values from 0 through 7.
<code>learn-vlan-1p-priority-except number</code>	(MX Series routers only) Do not match on the IEEE 802.1p learned VLAN priority field. Specify a single value or multiple values from 0 through 7.
<code>learn-vlan-id number</code>	(MX Series routers only) VLAN identifier used for MAC learning.
<code>learn-vlan-id-except number</code>	(MX Series routers only) Do not match on the VLAN identifier used for MAC learning.

Table 29: VPLS Firewall Filter Match Conditions (*continued*)

Match Condition	Description
<code>loss-priority level</code>	<p>Packet loss priority (PLP) level. Specify a single level or multiple levels: low, medium-low, medium-high, or high.</p> <p>Supported on MX Series routers; M120 and M320 routers; and M7i and M10i routers with the Enhanced CFEB (CFEB-E).</p> <p>On M320 routers, you must enable the tricolor statement at the [edit class-of-service] hierarchy level to commit a PLP configuration with any of the four levels specified. If the tricolor statement is not referenced, you can only configure the high and low levels. This applies to all protocol families.</p> <p>For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see the <i>JUNOS Class of Service Configuration Guide</i>.</p>
<code>loss-priority-except level</code>	<p>Do not match on the packet loss priority level. Specify a single level or multiple levels: low, medium-low, medium-high, or high.</p> <p>For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see the <i>JUNOS Class of Service Configuration Guide</i>.</p>
<code>port number</code>	(MX Series routers only) TCP or UDP source or destination port. You cannot specify both the port match condition and either the destination-port or source-port match condition in the same term.
<code>port-except number</code>	(MX Series routers only) Do not match on the TCP or UDP source or destination port. You cannot specify both the port match condition and either the destination-port or source-port match condition in the same term.
<code>source-mac-address address</code>	Source MAC address of a VPLS packet.
<code>source-port number</code>	(MX Series routers only) TCP or UDP source port field. You cannot specify the port and source-port match conditions in the same term.
<code>source-port-except number</code>	(MX Series routers only) Do not match on the TCP or UDP source port field. You cannot specify the port and source-port match conditions in the same term.
<code>tcp-flags flags</code>	<p>(MX Series routers only) One or more of the following TCP flags:</p> <ul style="list-style-type: none"> ■ Bit-name: fin, syn, rst, push, ack, urgent ■ Numerical value: 0x01 through 0x20 ■ Text synonym: tcp-established, tcp-initial <p>You can string together multiple flags using logical operators.</p> <p>Configuring the tcp-flags match condition requires that you configure the next-header-tcp match condition.</p>
<code>traffic-type type-name</code>	(MX Series routers only) Traffic type. Specify broadcast , multicast , unknown-unicast , or known-unicast .
<code>traffic-type-except type-name</code>	(MX Series routers only) Do not match on the traffic type. Specify broadcast , multicast , unknown-unicast , or known-unicast .
<code>user-vlan-1p-priority number</code>	IEEE 802.1p user priority field. Specify a single value or multiple values from 0 through 7.

Table 29: VPLS Firewall Filter Match Conditions (continued)

Match Condition	Description
<code>user-vlan-1p-priority-except number</code>	Do not match on the IEEE 802.1p user priority field. Specify a single value or multiple values from 0 through 7.
<code>user-vlan-id number</code>	(MX Series routers only) First VLAN identifier that is part of the payload.
<code>user-vlan-id-except number</code>	(MX Series routers only) Do not match on the first VLAN identifier that is part of the payload.
<code>vlan-ether-type value</code>	VLAN Ethernet type field of a VPLS packet.
<code>vlan-ether-type-except value</code>	Do not match on the VLAN Ethernet type field of a VPLS packet.

Related Topics ■ How to Specify Firewall Filter Match Conditions on page 202

Configuring Layer 2 Bridging Match Conditions for MX Series Ethernet Services Routers

Table 30 on page 197 describes the firewall filter match conditions supported for Layer 2 bridging traffic on MX Series routers.

To configure firewall filter match conditions for Layer 2 bridging traffic:

- Include the *match-conditions* statement at the [edit firewall family bridge filter *filter-name* term *term-name* from] hierarchy level.

Table 30: Layer 2 Bridging Firewall Filter Match Conditions (MX Series Ethernet Services Routers Only)

Match Condition	Description
<code>destination-mac-address address</code>	Destination media access control (MAC) address of a Layer 2 packet in a bridging environment.
<code>destination-port number</code>	TCP or UDP destination port field. You cannot specify both the <code>port</code> and <code>destination-port</code> match conditions in the same term.
<code>dscp number</code>	Differentiated Services code point (DSCP). The DiffServ protocol uses the type-of-service (ToS) byte in the IP header. The most significant 6 bits of this byte form the DSCP. For more information, see the <i>JUNOS Class of Service Configuration Guide</i> . You can specify the DSCP in hexadecimal, binary, or decimal form.
<code>ether-type value</code>	Ethernet type field of a Layer 2 packet in a bridging environment.
<code>ether-type-except value</code>	Do not match on the Ethernet type field of a Layer 2 packet.
<code>forwarding class class</code>	Forwarding class. Specify <code>assured-forwarding</code> , <code>best-effort</code> , <code>expedited-forwarding</code> , or <code>network-control</code> .
<code>forwarding-class-except class</code>	Ethernet type field of a Layer 2 packet environment. Specify <code>assured-forwarding</code> , <code>best-effort</code> , <code>expedited-forwarding</code> , or <code>network-control</code> .

Table 30: Layer 2 Bridging Firewall Filter Match Conditions (MX Series Ethernet Services Routers Only) (continued)

Match Condition	Description
<code>icmp-code number</code>	ICMP code field. The value or keyword provides more specific information than <code>icmp-type</code> . Because the value's meaning depends on the associated <code>icmp-type</code> , you must specify <code>icmp-type</code> along with <code>icmp-code</code> .
<code>icmp-type number</code>	ICMP packet type field. Normally, you specify this match in conjunction with the <code>protocol</code> match statement to determine which protocol is being used on the port.
<code>interface-group group-number</code>	Interface group on which the packet was received. An interface group is a set of one or more logical interfaces. For <code>group-number</code> , specify a value from 0 through 255.
<code>interface-group-except number</code>	Do not match on the interface group on which the packet was received.
<code>ip-address address</code>	32-bit address that supports the standard syntax for IPv4 addresses.
<code>ip-destination-address address</code>	32-bit address that is the final destination node address for the packet.
<code>ip-precedence ip-precedence-field</code>	IP precedence field. In place of the numeric field value, you can specify one of the following text synonyms (the field values are also listed): <code>critical-ecp</code> (0xa0), <code>flash</code> (0x60), <code>flash-override</code> (0x80), <code>immediate</code> (0x40), <code>internet-control</code> (0xc0), <code>net-control</code> (0xe0), <code>priority</code> (0x20), or <code>routine</code> (0x00).
<code>ip-precedence-except</code>	Do not match on the IP precedence field.
<code>ip-protocol number</code>	IP protocol field.
<code>ip-source-address address</code>	IP address of the source node sending the packet.
<code>learn-vlan-1p-priority value</code>	(Supported with bridging, VPLS, and Layer 2 circuit cross-connect [CCC] traffic only) IEEE 802.1p learned VLAN priority field. Specify a single value or multiple values from 0 through 7.
<code>learn-vlan-1p-priority-except value</code>	(Supported with bridging, VPLS, and Layer 2 circuit cross-connect [CCC] traffic only) Do not match on the IEEE 802.1p learned VLAN priority field. Specify a single value or multiple values from 0 through 7.
<code>learn-vlan-id number</code>	VLAN identifier used for MAC learning.
<code>learn-vlan-id-except number</code>	Do not match on the VLAN identifier used for MAC learning.
<code>loss-priority level</code>	<p>Packet loss priority (PLP) level. Specify a single level or multiple levels: <code>low</code>, <code>medium-low</code>, <code>medium-high</code>, or <code>high</code>.</p> <p>For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see the <i>JUNOS Class of Service Configuration Guide</i>.</p>
<code>loss-priority-except level</code>	<p>Do not match on the packet loss priority level. Specify a single level or multiple levels: <code>low</code>, <code>medium-low</code>, <code>medium-high</code>, or <code>high</code>.</p> <p>For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see the <i>JUNOS Class of Service Configuration Guide</i>.</p>

Table 30: Layer 2 Bridging Firewall Filter Match Conditions (MX Series Ethernet Services Routers Only) (continued)

Match Condition	Description
<i>port number</i>	TCP or UDP source or destination port. You cannot specify both the port match condition and either the destination-port or source-port match conditions in the same term.
<i>source-mac-address address</i>	Source MAC address of a Layer 2 packet.
<i>source-port number</i>	TCP or UDP source port field. You cannot specify the port and source-port match conditions in the same term.
<i>tcp-flags flags</i>	<p>One or more of the following TCP flags:</p> <ul style="list-style-type: none"> ■ Bit-name: fin, syn, rst, push, ack, urgent ■ Numerical value: 0x01 through 0x20 ■ Text synonym: tcp-established, tcp-initial <p>You can string together multiple flags using logical operators.</p> <p>Configuring the tcp-flags match condition requires that you configure the next-header-tcp match condition.</p>
<i>traffic-type type</i>	Traffic type. Specify broadcast , multicast , unknown-unicast , or known-unicast .
<i>traffic-type-except type</i>	Do not match on the traffic type.
<i>user-vlan-1p-priority value</i>	(Supported with bridging, VPLS, and Layer 2 CCC traffic only) IEEE 802.1p user priority field. Specify a single value or multiple values from 0 through 7.
<i>user-vlan-1p-priority-except value</i>	(Supported with bridging, VPLS, and Layer 2 CCC traffic only) Do not match on the IEEE 802.1p user priority field. Specify a single value or multiple values from 0 through 7.
<i>user-vlan-id number</i>	First VLAN identifier that is part of the payload.
<i>user-vlan-id-except number</i>	Do not match on the first VLAN identifier that is part of the payload.
<i>vlan-ether-type value</i>	VLAN Ethernet type field of a Layer 2 bridging or VPLS packet.
<i>vlan-ether-type-except value</i>	Do not match on the VLAN Ethernet type field of a Layer 2 bridging or VPLS packet.
Related Topics	<ul style="list-style-type: none"> ■ How to Specify Firewall Filter Match Conditions on page 202 ■ Overview of Protocol Match Conditions on page 200

Overview of Protocol Match Conditions

In a standard firewall filter, if you specify a port match condition or a match of the ICMP type, ICMP code, or TCP flags field or the TCP establish or TCP initial match conditions, there is no implied protocol match. If you use one of the following match conditions in a term, you should also explicitly specify the protocol as a match condition in the same term:

- **destination-port**—For IPv4, specify the match **protocol tcp** or **protocol udp** in the same term. For IPv6, specify the match **next-header tcp** or **next-header udp** in the same term.
- **icmp-code**—For IPv4, specify the match **protocol icmp** in the same term. For IPv6, specify the match **next-header icmp** or **next-header icmp6** in the same term.
- **icmp-type**—For IPv4, specify the match **protocol icmp** in the same term. For IPv6, specify the match **next-header icmp** or **next-header icmp6** in the same term.
- **port**—For IPv4, specify the match **protocol tcp** or **protocol udp** in the same term. For IPv6, specify the match **next-header tcp** or **next-header udp** in the same term.
- **source-port**—For IPv4, specify the match **protocol tcp** or **protocol udp** in the same term. For IPv6, specify the match **next-header tcp** or **next-header udp** in the same term.
- **tcp-established**—For IPv4, specify the match **protocol tcp** in the same term. For IPv6, specify the match **next-header tcp** in the same term.
- **tcp-flags**—For IPv4, specify the match **protocol tcp** in the same term. For IPv6, specify the match **next-header tcp** in the same term.
- **tcp-initial**—For IPv4, specify the match **protocol tcp** in the same term. For IPv6, specify the match **next-header tcp** in the same term.

When examining match conditions, the JUNOS Software tests only the specified field itself. The software does not also test the IP header to determine that the packet is indeed an IP packet.

If you do not explicitly specify the protocol, when using the fields listed previously, design your filters carefully to ensure that they are performing the expected matches. For example, if you specify a match of **destination-port ssh**, the JUNOS Software deterministically matches any packets that have a value of 22 in the 2-byte field that is 2 bytes beyond the end of the IP header, without ever checking the IP protocol field.

Example: Matching on Destination Port and Protocol Fields

The first term matches all packets except for TCP and UDP packets, so only TCP and UDP packets are evaluated by the third term (term **test-a-port**):

```
[edit]
firewall {
  family inet {
    filter test-filter {
```



```

term all-but-tcp-and-udp {
    from {
        protocol-except [tcp udp];
    }
    then accept;
}
term test-an-address {
    from {
        address 192.168/16;
    }
    then reject;
}
term test-a-port {
    from {
        destination-port [ssh dns];
    }
    then accept;
}
term dump-everything-else {
    then reject;
}
}
}
}

```

Overview of Class-Based Match Conditions

Class-based filter conditions match packet fields based on source class or destination class. A source class is a set of source prefixes grouped together and given a class name. A destination class is a set of destination prefixes grouped together and given a class name.

Use the `source-class class-name` statement to match on one or more source classes

Use the `destination-class class-name` statement to match on one or more destination classes.



NOTE: Both match conditions are supported for IPv4 and IPv6 traffic.

You can specify the destination class in the following ways:

- Destination-class usage (DCU) enables you can track how much traffic is sent to a specific prefix in the core of the network originating from one of the specified interfaces. However, DCU limits your ability to keep track of traffic moving in the reverse direction. It can account for all traffic that arrives on a core interface and heads toward a specific customer, but it cannot count traffic that arrives on a core interface from a specific prefix.
- Source-class usage (SCU) enables you to monitor the amount of traffic originating from a specific prefix. With this feature, usage can be tracked and customers can be billed for the traffic they receive

You can specify a source class or destination class for an output firewall filter. Although you can specify a source class and destination class for an input firewall filter, the counters are incremented only if the firewall filter is applied on the output interface.

The class-based filter match condition works only for output filters, because the SCU and DCU are determined after route lookup.



NOTE: SCU and DCU are not supported on the interfaces you configure as the output interface for tunnel traffic for transit packets exiting the router through the tunnel.

For more information about SCU and DCU, see the *Source Class Usage Feature Guide*.

How to Specify Firewall Filter Match Conditions

Because firewall filter match conditions can match a variety of criteria, including packet fields and IP addresses, you can specify the following types of values in a single match condition:

- Numeric value or range of values
- Single text value or multiple text values
- Multiple numeric and text values
- Single prefix value or multiple prefix values
- Single bit-field value
- Multiple bit-field values using logical operators

This topic covers:

- Numeric and Text Values in Match Conditions on page 202
- Prefixes in Match Conditions on page 203
- Bit-Field Values in Match Conditions on page 206

Numeric and Text Values in Match Conditions

Numeric Values

You can specify numeric values in one of the following ways:

- Single number. A match occurs if the value of the field matches the number. For example:

source-port 25;

- Range of numbers. A match occurs if the value of the field falls within the specified range. The following example matches source ports 1024 through 65,535, inclusive:

source-port 1024-65535;

Text Values

You can specify a text value as a synonym for a numeric value:

Text synonym for a single number. A match occurs if the value of the field matches the number that corresponds to the synonym. For example:

```
source-port smtp;
```

A match occurs if the value of the field is 25 since that numeric value corresponds to the text synonym, `smtp`.

Multiple Numeric and Text Values

To specify multiple values in a single match condition, group the values within square brackets following the keyword. A match occurs if the value of the field matches the number that corresponds to either of the text synonyms or any of the configured numerical values. For example:

```
source-port [ smtp ftp-data 25 1024-65535 ];
```

A match occurs if the value of the field in a packet matches any of the following values: 20 (since it corresponds to the text synonym `ftp-data`; 25 since it matches the text synonym `smtp`); or any value from 1024 through 65535.

Prefixes in Match Conditions

Address filter conditions match prefix values in a packet, such as IP source and destination prefixes. For address filter match conditions, you specify a keyword, such as `destination-address` or `destination-prefix`, that identifies the field and one or more prefixes of that type that a packet must match.

You can specify the address in one of the following ways:

- **Single prefix**—A match occurs if the value of the field matches the prefix. For example:

```
[edit firewall family family-name filter filter-name term term-name from]
destination-address 10.0.0.0/8;
```

In this example, a match occurs if a destination address matches the prefix 10.0.0.0/8

- **Multiple prefixes**—A match occurs if any one of the prefixes in the list matches the packet. For example:

```
[edit firewall family family-name filter filter-name term term-name from]
destination-address {
  10.0.0.0/8;
  192.168.0.0/32;
}
```

In this example, a match occurs if a destination address matches either the 10.0.0.0/8 or the 192.168.0.0/32 prefix.

To exclude a prefix, specify the string **except** after the prefix. In the following example, any addresses that fall under the **192.168.10.0/8** prefix match, except for addresses that fall under **192.168.0.0/16**. All other addresses implicitly do not match this condition.

```
[edit firewall family family-name filter filter-name term term-name from]
destination-address {
  192.168.0.0/16 except;
  192.168.10.0/8;
}
```

To match all destinations except one, in this example **10.1.1.0/24**, configure the match conditions as follows:

```
[edit firewall family family-name filter filter-name term term-name from]
destination-address {
  0.0.0.0/0;
  10.1.1.0/24 except;
}
```

To specify the address prefix, use the notation *prefix/prefix-length*. If you do not specify *prefix-length*, it defaults to **/32**, as shown in this example:

```
[edit firewall family family-name filter filter-name term term-name from]
user@host# set destination-address 10
[edit firewall family family-name filter filter-name term term-name from]
user@host# show
destination-address {
  10.0.0.0/32;
}
```

You can also specify a netmask value rather than a prefix length, for example:

```
[edit firewall family inet filter filter-name]
term term-name {
  address 10.0.0.10/255.0.0.255;
}
```

Noncontiguous Address Prefixes

You can specify noncontiguous address prefixes in a filter term for firewall filters. Noncontiguous address prefixes are prefixes that are not adjacent or neighboring to one another. For example, in the following example, the following prefixes are noncontiguous: **0.0.0.10/0.0.0.255**, **0.10.0.10/0.255.0.255**, and **0.12.10.9/0.255.255.255**:

```
[edit firewall family inet filter filter-name]
term term-name {
  address 0.0.0.10/0.0.0.255;
  destination-address 0.10.0.10/0.255.0.255;
  source-address 0.12.10.9/0.255.255.255 except;
}
```



NOTE: Noncontiguous address prefixes are valid only for IPv4 filters. IPv6 filters do not support noncontiguous address prefixes.

The prefix notation shown matches any address with a first and last octet of 10. The address and netmask are separated by a forward slash (/). The second and third bytes of the prefix can be any value from 0 through 255.

Prefix Order

The order in which you list prefixes in the list is not significant. They are all evaluated to determine whether a match occurs. If prefixes overlap, longest-match rules are used to determine whether a match occurs. Each list of prefixes contains an implicit 0/0 **except** statement, which means that any prefix that does not match any prefix in the list is explicitly considered not to match.

Because the prefixes are order-independent and use longest-match rules, longer prefixes subsume shorter ones as long as they are the same type (whether you specify **except** or not). This is because anything that would match the longer prefix would also match the shorter one. Consider the following example:

```
[edit firewall family family-name filter filter-name term term-name from]
source-address {
  172.16.0.0/10;
  172.16.2.0/16 except;
  192.168.1.0;
  192.168.1.192/26 except;
  192.168.1.254;
  172.16.3.0/16; # ignored
  0.0.0.0/0 except; # ignored
}
```

- 172.16.1.2 matches the 172.16.0.0/10 prefix, and thus the action in the **then** statement is taken.
- 172.16.2.2 matches the 172.16.2.0/16 prefix. Because this prefix is negated (that is, marked as **except**), an explicit mismatch occurs. The next term in the filter is evaluated, if there is one. If there are no more terms, the packet is discarded.
- 10.1.2.3 does not match any of the prefixes included in the **source-address** condition. Instead, it matches the implicit 0.0.0.0/0 **except** at the end of the list, and is considered to be a mismatch.
- The 172.16.3.0/16 statement is ignored because it falls under the address 172.16.0.0/10—both are the same type.
- The 10.2.2.2 **except** statement is ignored because it is subsumed by the implicit 0.0.0.0/0 **except** statement at the end of the list.



BEST PRACTICE: When a firewall filter term includes the **from address** *address* match condition and a subsequent term includes the **from source-address** *address* match

condition for the same address, packets may be processed by the latter term before they are evaluated by any intervening terms. Therefore, packets that should be rejected by the intervening terms may be accepted, or packets that should be accepted may be rejected.

To prevent this from occurring, we recommend that you do the following. For every firewall filter term that contains the **from address address** match condition, replace that term with two separate terms: one that contains the **from source-address address** match condition, and another that contains the **from destination-address address** match condition.

Prefix Lists

You can also define a list of IP address prefixes under a *prefix-list* alias for frequent reference. You make this definition at the [edit policy-options] hierarchy level:

```
[edit policy-options]
policy-options {
  prefix-list prefix-list {
    address;
    address;
    address;
  }
}
```

After you have defined a prefix list, you can use it when defining firewall filters:

```
[edit firewall family family-name filter filter-name term term-name]
from {
  source-prefix-list {
    prefix-list1;
    prefix-list2;
  }
  destination-prefix-list {
    prefix-list1;
  }
}
```

Bit-Field Values in Match Conditions

Bit-field filter conditions match packet fields if particular bits in those fields are or are not set.

The following conditions match on bit-field values:

- first-fragment
- fragment-flags
- is-fragment
- tcp-established

- tcp-flags
- tcp-initial



NOTE: The JUNOS Software does not automatically check the first fragment bit when matching TCP flags. For IPv4 traffic only, to include the first fragment bit, include the **fragment-offset** match condition. This condition is not supported for any other protocol family.

Single Bit-Field Value

To specify the bit-field value to match, enclose the value in quotation marks (“ ”). For example, a match occurs if the RST bit in the TCP flags field is set:

```
tcp-flags "rst";
```

Generally, you specify the bits being tested using text synonyms. Bit-field match text values always map to a single bit value. You also can specify bit fields as hexadecimal or decimal numbers.

To negate a match, precede the value with an exclamation point. For example, a match occurs only if the RST bit in the TCP flags field is *not* set:

```
tcp-flags "!rst";
```

Multiple Bit-Field Values

To match multiple bit-field values, use the logical operators list in Table 31 on page 207. The operators are listed in order, from highest precedence to lowest precedence. Operations are left-associative.

Table 31: Bit-Field Logical Operators

Logical Operator	Description
(...)	Grouping
!	Negation
& or +	Logical AND
or ,	Logical OR

As an example of a logical AND operation, in the following, a match occurs if the packet is the initial packet on a TCP session:

```
tcp-flags "syn & !ack";
```

In this example, a match occurs if the SYN flag is set. This flag is set only in the initial packet sent on a TCP session. A match does not occur if the ACK flag is set. The ACK flag is set in all packets sent after the initial packet.

As an example of a logical OR operation, in the following, a match occurs if the packet is *not* the initial packet on a TCP session:

```
tcp-flags "!syn | ack";
```

In this example, a match occurs either if the SYN flag is not set or if the ACK flag is set. Because the SYN flag is set only in the initial packet sent on a TCP session and the ACK flag is set in all packets sent after the initial packet, a match occurs if the packet is *not* the initial packet.

As an example of grouping, in the following, a match occurs for any packet that is either a TCP reset or is not the initial packet in the session:

```
tcp-flags "!(syn & !ack) | rst";
```

In this example, a match occurs if the SYN flag is not set and the ACK field is set or if the RST field is set. Because the SYN flag is set only in the initial packet sent on a TCP session and the ACK flag is set in all packets sent after the initial packet, a match occurs if the packet is *not* the initial packet. A match also occurs if the packet has the TCP reset flag set.

When you specify a numeric value that has more than one bit set, the value is treated as a logical AND of the set bits. For example, the following two values are the same and a match occurs only if either bit 0x01 or 0x02 is not set:

```
tcp-flags "!0x3";
tcp-flags "!(0x01 & 0x02)";
```

You can use text synonyms to specify some common bit-field matches. You specify these matches as a single keyword. For example:

```
tcp-established;
```

The `tcp-established` condition matches on TCP packets other than the first packet of a connection. This condition is a synonym for `"(ack | rst)"`.

Related Topics

Configuring Actions in Firewall Filter Terms

In the `then` statement in a firewall filter term, you specify the actions to perform on packets whose characteristics match the conditions specified in the preceding `from` statement. To configure a filter action, include the `then` statement at the `[edit firewall family family-name filter filter-name term term-name]` hierarchy level:

```
[edit firewall family family-name filter filter-name term term-name]
then {
    action;
    nonterminating actions;
}
```




BEST PRACTICE: We strongly recommend that you always explicitly configure an action in the **then** statement. If you do not, or if you omit the **then** statement entirely, packets that match the conditions in the **from** statement are accepted.

You can specify only one filter terminating action statement (or omit it), but you can specify any combination of nonterminating actions. For the action or nonterminating action to take effect, all conditions in the **from** statement must match. If you specify **log** as one of the actions in a term, this constitutes a terminating action; whether any additional terms in the filter are processed depends on the traffic through the filter.

The nonterminating action operations carry a default **accept** action. For example, if you specify a nonterminating action and do not specify an action, the specified nonterminating action is implemented and the packet is accepted. To circumvent an implicit **accept** action and allow the JUNOS Software to evaluate the following term in the filter, use the **next term** statement.

The following actions are terminating actions:

- **accept**
- **discard**
- **reject**
- **logical-system** *logical-system-name*
- **routing-instance** *routing-instance-name*
- **topology** *topology-name*



NOTE: You cannot configure the **next term** action with a terminating action in the same filter term. You can only configure the **next term** action with another nonterminating action in the same filter term.

Policing uses a specific type of action, known as a policer action. For more information, see “Policer Configuration” on page 257.

For more information about forwarding classes and loss priority, see the *JUNOS Class of Service Configuration Guide*.

Table 32 on page 209 shows the complete list of filter actions, both terminating and non-terminating.

Table 32: Firewall Filter Actions

Action	Description
accept	Accept a packet.
count <i>counter-name</i>	Count the packet in the specified counter.

Table 32: Firewall Filter Actions (*continued*)

Action	Description
dscp	(Family <code>inet</code> only) Set the IPv4 Differentiated Services code point (DSCP) bit to 0.
discard	Discard a packet silently, without sending an Internet Control Message Protocol (ICMP) message. Discarded packets are available for logging and sampling.
forwarding-class class	Classify the packet into one of the following forwarding classes: <code>as</code> , <code>assured-forwarding</code> , <code>best-effort</code> , <code>expedited-forwarding</code> , or <code>network-control</code> .
ipsec-sa <i>ipsec-sa</i>	(Family <code>inet</code> only) Use the specified IPsec security association. NOTE: This action is not support on MX Series routers.
load-balance group-name	(Family <code>inet</code> only) Use the specified load-balancing group.
log	(Family <code>inet</code> and <code>inet6</code> only) Log the packet header information in a buffer within the Packet Forwarding Engine. You can access this information by issuing the <code>show firewall log</code> command at the command-line interface (CLI).
logical-system logical-system-name	Specify a logical system to which packets are forwarded.
loss-priority (high medium-high medium-low low)	Set the loss priority level for packets. Supported on MX Series routers; M120 and M320 routers; and M7i and M10i routers with the Enhanced CFEB (CFEB-E). On M320 routers, you must enable the <code>tricolor</code> statement at the <code>[edit class-of-service]</code> hierarchy level to commit a PLP configuration with any of the four levels specified. If the <code>tricolor</code> statement is not referenced, you can only configure the <code>high</code> and <code>low</code> levels. This applies to all protocol families. You cannot also configure the <code>three-color-policer</code> action modifier for the same firewall filter term. These two action modifiers are mutually exclusive.
next term	Continue to the next term for evaluation.
next-hop-group group-name	(Family <code>inet</code> only) Use the specified next-hop group.
policer policer-name	Rate-limit packets based on the specified policer.
port-mirror	(Family <code>bridge</code> , <code>ccc</code> , <code>inet</code> , <code>inet6</code> , and <code>vpls</code> only) Port-mirror packets based on the specified family. Supported on M120 routers, M320 routers configured with Enhanced III FPCs, and MX Series routers only.
prefix-action <i>name</i>	(Family <code>inet</code> only) Count or police packets based on the specified action name.

Table 32: Firewall Filter Actions (*continued*)

Action	Description
<code>reject</code> <i>message-type</i>	Discard a packet, sending an ICMPv4 or an ICMPv6 destination unreachable message. Rejected packets can be logged or sampled if you configure either the sample or the syslog action modifier. You can specify one of the following message codes: administratively-prohibited (default), bad-host-tos , bad-network-tos , host-prohibited , host-unknown , host-unreachable , network-prohibited , network-unknown , network-unreachable , port-unreachable , precedence-cutoff , precedence-violation , protocol-unreachable , source-host-isolated , source-route-failed , or tcp-reset . If you specify tcp-reset , a Transmission Control Protocol (TCP) reset is returned if the packet is a TCP packet. Otherwise, the default code of administratively-prohibited , which has a value of 13, is returned. Supported for family inet and inet6 only.
<code>routing-instance</code> <i>routing-instance</i>	(Family inet and inet6 only) Specify a routing instance to which packets are forwarded.
<code>sample</code>	(Family inet , inet6 , and mpls only) Sample the packets.
<code>syslog</code>	Log the packet to the system log file.
<code>three-color-policer</code> <i>policer-name</i>	Apply rate limits to the traffic using the tricolor marking policer. You cannot also configure the loss-priority action modifier for the same firewall filter term. These two action modifiers are mutually exclusive.
<code>topology</code> <i>topology-name</i>	(Family inet and inet6 only) Specify a topology to which packets are forwarded.

Example: Counting and Sampling Accepted Packets

Count, sample, and accept the traffic:

```
term all {
  then {
    count sam-1;
    sample; # default action is accept
  }
}
```

Display the packet counter:

```
user@host> show firewall filter sam
Filter:
Counters:
Name           Bytes           Packets
sam
sam-1          98              8028
```

Display the firewall log output:

```
user@host> show firewall log
Time   Filter  A Interface  Pro Source address  Destination address
23:09:09 -      A at-2/0/0.301  TCP 10.2.0.25      10.211.211.1:80
```

```

23:09:07 -      A at-2/0/0.301      TCP 10.2.0.25      10.211.211.1:56
23:09:07 -      A at-2/0/0.301      ICM 10.2.0.25      10.211.211.1:49552
23:02:27 -      A at-2/0/0.301      TCP 10.2.0.25      10.211.211.1:56
23:02:25 -      A at-2/0/0.301      TCP 10.2.0.25      10.211.211.1:80
23:01:22 -      A at-2/0/0.301      ICM 10.2.2.101     10.211.211.1:23251
23:01:21 -      A at-2/0/0.301      ICM 10.2.2.101     10.211.211.1:16557
23:01:20 -      A at-2/0/0.301      ICM 10.2.2.101     10.211.211.1:29471
23:01:19 -      A at-2/0/0.301      ICM 10.2.2.101     10.211.211.1:26873

```

This output file contains the following fields:

- **Time**—Time at which the packet was received (not shown in the default).
- **Filter**—Name of a filter that has been configured with the **filter** statement at the [edit firewall] hierarchy level. A hyphen (-) or the abbreviation **pfe** indicates that the packet was handled by the Packet Forwarding Engine. A space (no hyphen) indicates that the packet was handled by the Routing Engine.
- **A**—Filter action:
 - **A**—Accept (or next term)
 - **D**—Discard
 - **R**—Reject
- **Interface**—Interface on which the filter is configured.



NOTE: We strongly recommend that you always explicitly configure an action in the **then** statement.

- **Pro**—Packet's protocol name or number.
- **Source address**—Source IP address in the packet.
- **Destination address**—Destination IP address in the packet.

Display the sampling output:

```
user@host> show log /var/tmp/sam
```

```
# 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



NOTE: When you enable reverse path forwarding (RPF) on an interface with an input filter for firewall log and count, the input firewall filter does not log the packets rejected by RPF, although the rejected packets are counted. To log the rejected packets, use an RPF check fail filter.

For more information about sampling output, see “Applying Filters to Forwarding Tables” on page 329.

Example: Setting the DSCP Bit to Zero

Set the DSCP bit to 0 (zero) using a firewall filter:

```
firewall {
  filter filter1 {
    term 1 {
      from {
        dscp 2;
      }
      then {
        dscp 0;
        forwarding-class best-effort;
      }
    }
    term 2 {
      from {
        dscp 3;
      }
      then {
        forwarding-class best-effort;
      }
    }
  }
}
```

Apply this filter to the logical interface corresponding to the VPN routing and forwarding (VRF) instance:

```
interfaces so-0/1/0 {
  unit 0 {
    family inet {
      filter input filter1;
    }
  }
}
```

Configuring Nested Firewall Filters

You can configure a filter within the term of another filter to minimize the work needed to configure terms common to numerous filters. Each firewall filter consists of one or more *terms*. You can configure one filter with the common desired terms, and apply them to other filters. To make changes to the common desired terms, you need to make term modifications only to the filter with the common terms instead of changing terms on every filter.

To configure a filter within a filter, include the `filter` statement at the `[edit firewall filter inet filter-name term term-name]` hierarchy level:

```
term term-name {
  filter filter-name;
```

```
}
```

A filter within a filter cannot reference yet another filter. For example, the following configuration is *not* valid:

```
[edit]
firewall {
  filter filter-name {
    term t1 {
      filter filter-name2 {
        term t2 {
          filter filter-name3;
        }
      }
    }
  }
}
```

You cannot configure the **from** or **then** statement under the same filter term that references a filter within a filter. For example, the following configuration is *not* valid:

```
[edit]
firewall {
  filter filter-name {
    term t1 {
      filter filter-name2 {
        then {
          accept;
        }
      }
    }
  }
}
```

The maximum number of filters within a filter is limited to 256.

Example: Configuring Nested Filters

Define a filter `common-filter` and configure it into two separate filters:

```
[edit]
firewall {
  filter common-filter {
    term t1 {
      from {
        protocol udp;
        port tftp;
      }
      then {
        log;
        discard;
      }
    }
  }
  filter filter1 {
```

```

        term term1 {
            filter common-filter;
        }
    }
    filter filter2 {
        term term1 {
            filter common-filter;
        }
    }
}

```

Applying Firewall Filters to Interfaces

For a firewall filter to work, you must apply it to at least one interface. To do this, include the `filter` statement when configuring the logical interface at the `[edit interfaces interface-name unit logical-unit-number family family-name]` hierarchy level:

```

[edit interfaces interface-name unit logical-unit-number family family-name]
filter {
    input filter-name;
    input-list [ filter-names ];
    output filter-name;
    output-list [ filter-names ];
}

```

In the `input` statement, list the name of one firewall filter to be evaluated when packets are received on the interface. Input filters applied to the loopback interface, `lo0`, affect only inbound traffic destined for the Routing Engine.

In the `output` statement, list the name of one firewall filter to be evaluated when packets are transmitted on the interface. Output filters applied to the loopback interface, `lo0`, affect only outbound traffic sent from the Routing Engine.



NOTE: On MX Series routers only, you cannot apply as an output filter, a firewall filter configured at the `[edit firewall filter family ccc]` hierarchy level. Firewall filters configured for the `family ccc` statement can be applied only as input filters on MX Series routers.

In the `input-list` statement, list the names of firewall filters to be evaluated when packets are received on the interface. You can specify up to 16 firewall filters for the filter input list. In the `output-list` statement, list the names of firewall filters to be evaluated when packets are transmitted from the interface. You can specify up to 16 firewall filters for the filter output list.

Unless you use an `input-list` or an `output-list`, you can apply only one input and one output firewall filter to each interface. You can use the same filter one or more times.

The `input-list` and `output-list` statements are not supported for simple filters or service filters. For more information about applying input lists and output lists, see “Overview of Firewall Filter Lists” on page 219.

For more general information about configuring filters on interfaces, see the *JUNOS Network Interfaces Configuration Guide*.

When you apply a filter to an interface, it is evaluated against all the data packets passing through that interface. The exception is the loopback interface, **lo0**, which is the interface to the Routing Engine and carries no data packets. If you apply a filter to the **lo0** interface, the filter affects the local packets received or transmitted by the Routing Engine.

Filters apply to all packets entering an interface, not just the packets destined for the Routing Engine. To filter packets destined for the Routing Engine, configure the **group** statement at the [edit interfaces *interface-name* unit *logical-unit-number* family *family-name* filter] hierarchy level. For more information, see “Defining Interface Groups” on page 217.

You can configure the following additional properties when applying filters to interfaces:

- Configuring Interface-Specific Counters on page 216
- Defining Interface Groups on page 217

Configuring Interface-Specific Counters

When you configure a firewall filter that is applied to multiple interfaces, you can name individual counters specific to each interface. These counters enable you to easily maintain statistics on the traffic transiting the different interfaces. A separate instance of the interface-specific firewall filter is created for each interface to which you apply the filter.



NOTE: Configuration of interface-specific counters also creates separate instances of any policers and counters you have configured for the same interface. For more information about policers, see “Policer Configuration” on page 257.

To configure interface-specific counters, include the **interface-specific** statement at the [edit firewall family *family-name* filter *filter-name*] hierarchy level:

```
[edit firewall family filter filter-name]
  interface-specific;
```



NOTE: The counter name is restricted to 24 bytes. If the renamed counter exceeds this maximum length, the policy framework software might reject it.

Example: Configuring Interface-Specific Counters

Configure an interface-specific counter:

```
[edit firewall]
family inet {
  filter test {
```



```

interface-specific;
term 1 {
  from {
    address {
      10.0.0.0/12;
    }
    protocol tcp;
  }
  then {
    count sample1;
    accept;
  }
}
}

```

When you apply this filter to the input interface of **at-1/1/1.0** and the output interface of **so-2/2/2.2**, the counters are named **sample1-at-1/1/1.0-i** and **sample1-so-2/2/2/.2-o**. The suffixes **-i** (input) and **-o** (output) are added to the counter names automatically.

The JUNOS Software 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.

Defining Interface Groups

When applying a firewall filter, you can define an interface to be part of an *interface group*. Packets received on that interface are tagged as being part of the group. You then can match these packets using the **interface-group** match statement, as described in “Configuring IPv4 Match Conditions” on page 183. The **interface-group** match statement is supported only by the IPv4, IPv6, circuit cross-connects (CCC), and VPLS protocol families.

To define an interface to be part of an interface group, include the **group** statement at the [edit interfaces *interface-name* unit *logical-unit-number* family *family-name* filter] hierarchy level:

```

[edit interfaces interface-name unit logical-unit-number family filter]
group group-number;
input filter-name;
output filter-name;

```

In the **group** statement, specify the interface group number to be associated with the filter.

In the **input** statement, list the name of one firewall filter to be evaluated when packets are received on the interface.

In the **output** statement, list the name of one firewall filter to be evaluated when packets are transmitted on the interface.



NOTE: The JUNOS Software also supports defining interface sets to which you can apply a firewall filter. An interface set lets you define a group a set of logical interfaces and apply hierarchical schedulers for class of services (CoS) to the interface set. For more information about the `interface-set` *interface-set-name* firewall filter match condition, see the “Configuring IPv4 Match Conditions” on page 183. The `interface-set` match condition is supported by the IPv4, IPv6, and protocol-independent protocol families and on MX Series routers only. For more information about configuring hierarchical schedulers for CoS, see the *JUNOS Class of Service Configuration Guide*.

Example: Defining Interface Groups

Create a filter that contains an interface group:

```
[edit firewall]
family inet {
  filter if-group {
    term group1 {
      from {
        interface-group 1;
        address {
          192.168.80.114/32;
        }
        protocol tcp;
        port finger;
      }
      then {
        count if-group-counter1;
        log;
        reject;
      }
    }
    term group-2 {
      then {
        count if-group-counter2;
        log;
        accept;
      }
    }
  }
}
```

Assign one or more interfaces to the interface group referenced in the filter:

```
[edit interfaces]
fxp0 {
  unit 0 {
    family inet {
      filter {
        group 1;
      }
      address 192.168.5.38/24;
    }
  }
}
```

```
}
```

Apply the filter that contains an interface group:

```
[edit interfaces]
lo0 {
  unit 0 {
    family inet {
      filter {
        input if-group;
        group 1;
      }
      address 10.0.0.1/32;
      address 192.168.77.1/32;
    }
  }
}
```

Overview of Firewall Filter Lists

Firewall filter lists effectively enable you to chain multiple firewall filters and apply them to a single interface. Typically, you apply a single firewall filter to an interface in the input or output direction or both. The ability to chain multiple firewall filters is useful when you have a router configured with many, even hundreds of interfaces, and you want to apply a unique filter to each interface but also apply a common set of terms to many or most of the interfaces on the router.

The most straightforward way to chain multiple firewall filter is to configure multiple, separate firewall filters. In such a scenario, you can configure multiple filters that are each unique to one interface and one or more separate filters that include the common terms that apply to many or most of the interface. You can then apply each unique filter only to the specific interface for which it is defined, along with the filter or filters that apply to many interfaces. This approach gives you the flexibility of being able to update a filter that applies only to one interface without having to update the configuration for all the other interfaces.

A second approach to chaining multiple firewall filters is to configure one or more filters within a filter, or *nested firewall filter*. To configure a nested firewall filter, you must first define each filter that you plan to nest by configuring it at the `[edit firewall]` hierarchy level. You then reference each filter you want to nest by including the `filter filter-name` statement at the `[edit firewall filter filter-name family family-name term term-name]` hierarchy level. You can then apply any combination of nested and standard firewall filters to interfaces as input lists or output lists. The advantage of this approach is that you can update any referenced firewall filter without having to update the nested firewall filter itself. Another advantage of nested firewall filters is that you can include a filter that you defined at the `[edit firewall]` hierarchy level in multiple nested filters.

In the following example, you configure multiple firewall filters, each of which is applied individually as part of an input list or an output list. Configuring multiple filters that include only one term enables you to update any one filter quickly without affecting any of the other filters.

```

[edit]
firewall {
  family inet {
    filter if1 {
      term 0 {
        from {
          destination-port 21;
        }
        then accept;
      }
    }
    filter if2 {
      term 0 {
        from {
          destination-port 23;
        }
        then accept;
      }
    }
    filter if3 {
      term 0 {
        from {
          destination-port 22;
        }
        then accept;
      }
    }
    filter of1 {
      term 0 {
        from {
          dscp af11;
        }
        then accept;
      }
    }
    filter of2 {
      term 0 {
        from {
          is-fragment;
        }
        then accept;
      }
    }
    filter of3 {
      term 0 {
        from {
          protocol ospf;
        }
        then accept;
      }
    }
  }
}

```

To apply the filters in this example on incoming and outbound traffic, use the `input-list` [*filter-names*] and `output-list` [*filter-names*] statements. In the following example, a list

of three input filters and a list of three output filters are applied to the `ge-1/3/0` interface. The filters are processed in the order in which they are applied.

```
[edit]
interfaces {
  ge-1/3/0 {
    unit 0 {
      family inet {
        filter {
          input-list [ if1 if2 if3 ];
          output-list [ of1 of2 of3 ];
        }
        address 1.1.1.2/30;
      }
    }
  }
}
```

Nested firewall filters also give you the ability to apply each filter within the filter in a multiple OR order. When you specify more than one match condition within a single term, both conditions (for example, source port and source address) must be met for a packet to match. In a nested firewall filter, a packet can match *either* the source port as defined in one filter within a filter, *or* the source address, as defined in another term or filter within a filter.

In contrast, in a standard firewall filter, multiple conditions within a single term are applied in a multiple AND order. If you specify more than one match condition within a single term, both conditions (for example, source port and source address) must be met for a packet to match.

An additional advantage of nested firewall filters is that if you need to update a specific filter within a filter, you can do so without having to update the nested filter itself.

The following example shows a nested firewall filter configuration. First, you define the Filter `f1` that you want to nest within a firewall filter. Then you reference Firewall Filter `f1` within the nested firewall filter, named `f2`. When you need to update Filter `f1`, you can do so without having to update Filter `f2`. The example also includes standard Filter `f3`, which you also apply as part of input list. You then apply firewall Filters `f2` and `f3` to interface `so-1/2/3` unit 0 as an input list. You do not need to apply filter `f1` directly to the interface because it is referenced in Filter `f2`.

■ Defining Filter f1

```
[edit firewall]
family inet {
  filter f1 {
    from {
      source-address 192.168.27.14
    }
    then count got-one
  }
}
```

■ Nesting Filter f1 in Filter f2

```
[edit firewall]
family inet {
  filter f2 {
    term 1
      filter f1; # Reference filter f1 defined at [edit the firewall hierarchy].
                # You must reference the filter within a term. Include only the name
                # of the filter you want to reference.
    }
    term 2 {
      from {
        source-port 3000;
      }
      then accept;
    }
  }
}
```

- Configuring standard firewall Filter f3

```
[edit firewall]
family inet {
  filter f3;
  term 3;
  from {
    icmp-code 3;
  }
  then accept;
}
```

- Applying Filters f2 and f3 as an input list

```
[edit interfaces so-1/2/3]
unit 0 {
  family inet {
    filter {
      input-list [ f2 f3 ]; # When you apply filter f2, it includes the referenced filter
                          # f1.
    }
  }
}
```

When you configure a list of firewall filters and apply them to an interface using either the **input-list** or **output-list** statement, the filters are concatenated into one consolidated and renamed firewall filter, and all policers and counters are also renamed. Each filter in the list is evaluated in the order in which it is applied to the interface.

The concatenated firewall filter is renamed based on the name of the interface and the direction in which the filter is applied:

- A list of firewall filters applied as input filters on interface **so-1/0/0 unit 0** becomes **so-1/0/0.0-i**.
- A list of firewall filters applied as output filters on interface **so-1/1/1 unit 0** becomes **so-1/1/1.0-o**.

Any counters or policers in the filter list are also renamed. The interface name and the letter *o* or *i* to indicate the direction of the filter are added to the end of the original counter name as follows:

- A counter named **bad-packets** in a filter applied in an output list to interface **so-2/2/0 unit 0** becomes **bad-packets-so-2/2/0.0-o**.
- A counter named **icmp-code-3** in a filter applied in an input list to interface **so-3/3/0 unit 0** becomes **icmp-code-3-so-3/3/0.0-i**.

Understanding how firewall filter lists are displayed is important when you use the **show firewall** command to view information about configured firewall filters. For example, the entry for a firewall filter named **c** that is applied in an output list to interface **at-1/0/1 unit 0** is displayed as **c-at-1/0/1.0-o**.

Related Topics

- Configuring Nested Firewall Filters on page 213
- Applying Firewall Filters to Interfaces on page 215

Firewall Filter Examples

The following examples illustrate how to define firewall filters:

- Example: Blocking Telnet and SSH Access on page 223
- Example: Blocking TFTP Access on page 224
- Example: Accepting DHCP Packets with Specific Addresses on page 225
- Example: Defining a Policer for a Destination Class on page 225
- Example: Counting IP Option Packets on page 226
- Example: Accepting OSPF Packets from Certain Addresses on page 227
- Example: Matching Packets Based on Two Unrelated Criteria on page 227
- Example: Counting Both Accepted and Rejected Packets on page 228
- Example: Blocking TCP Connections to a Certain Port Except from BGP Peers on page 228
- Example: Accepting Packets with Specific IPv6 TCP Flags on page 229
- Example: Setting a Rate Limit for Incoming Layer 2 Control Packets on page 230

Example: Blocking Telnet and SSH Access

Block telnet and SSH access to all but the **192.168.1.0/24** subnet. This filter also logs any SSH or telnet traffic attempts from other subnets to the firewall log buffer:

```
[edit]
firewall {
  family inet {
    filter local-access-control {
      term terminal-access {
        from {
```

```

        address {
            192.168.1.0/24;
        }
        protocol tcp;
        port [ssh telnet];
    }
    then accept;
}
term terminal-access-denied {
    from {
        protocol tcp;
        port [ssh telnet];
    }
    then {
        log;
        reject;
    }
}
term default-term {
    then accept;
}
}
}
}

```

Example: Blocking TFTP Access

Block Trivial File Transfer Protocol (TFTP) access, logging any attempts to establish TFTP connections:

```

[edit]
firewall {
    family inet {
        filter tftp-access-control {
            term one {
                from {
                    protocol udp;
                    port tftp;
                }
                then {
                    log;
                    discard;
                }
            }
        }
    }
}

```

By default, to decrease vulnerability to denial-of-service (DoS) attacks, the JUNOS Software filters and discards Dynamic Host Configuration Protocol (DHCP) or Bootstrap Protocol (BOOTP) packets that have a source address of 0.0.0.0 and a destination address of 255.255.255.255. This default filter is known as a unicast RPF check. However, some vendors' equipment automatically accepts these packets. To interoperate with other vendors' equipment, you can configure a filter that checks

for both these addresses and overrides the default RPF-check filter by accepting these packets.

Example: Accepting DHCP Packets with Specific Addresses

Configure a filter (`rpf-dhcp`) that accepts DHCP packets with a source address of 0.0.0.0 and a destination address of 255.255.255.255:

```
[edit firewall family inet]
filter rpf-dhcp {
  term dhcp {
    from {
      source-address {
        0.0.0.0/32;
      }
      destination-address {
        255.255.255.255/32;
      }
    }
    then {
      accept;
    }
  }
}
```

To apply this filter to an interface, include the `rpf-check fail-filter` statement at the `[edit interface interface-name unit logical-unit-number family family-name]` hierarchy level:

```
[edit interface interface-name unit logical-unit-number family inet]
rpf-check fail-filter rpf-dhcp;
```

Example: Defining a Policer for a Destination Class

Define a policer for destination class `class1`:

```
[edit]
firewall {
  family inet {
    filter filter1 {
      policer police-class1 {
        if-exceeding {
          bandwidth-limit 25;
          burst-size-limit 1000;
        }
        then {
          discard;
        }
      }
    }
    term term1 {
      from {
        destination-class class1;
      }
      then {
        policer police-class1;
      }
    }
  }
}
```

```

    }
  }
}

```

Example: Counting IP Option Packets

Count individual IP option packets, but do not block any traffic. Also, log packets that have loose or strict source routing:

```

[edit]
firewall {
  family inet {
    filter ip-option-filter {
      term match-strictsource {
        from {
          ip-options strict-source-route;
        }
        then {
          count strict-source-route;
          log;
          accept;
        }
      }
      term match-loose-source {
        from {
          ip-options loose-source-route;
        }
        then {
          count loose-source-route;
          log;
          accept;
        }
      }
      term match-record {
        from {
          ip-options record-route;
        }
        then {
          count record-route;
          accept;
        }
      }
      term match-timestamp {
        from {
          ip-options timestamp;
        }
        then {
          count timestamp;
          accept;
        }
      }
      term match-router-alert {
        from {

```

```

        ip-options router-alert;
    }
    then {
        count router-alert;
        accept;
    }
}
term match-all {
    then accept;
}
}
}
}

```

Example: Accepting OSPF Packets from Certain Addresses

Accept only OSPF packets from an address in the prefix 10.108.0.0/16, discarding all other packets with an administratively-prohibited ICMP message:

```

[edit]
firewall {
    family inet {
        filter ospf-filter {
            term term1 {
                from {
                    source-address {
                        10.108.0.0/16;
                    }
                    protocol ospf;
                }
            }
            term default-term {
                then {
                    reject administratively-prohibited; # default reject action
                }
            }
        }
    }
}

```

Example: Matching Packets Based on Two Unrelated Criteria

Match packets that are either OSPF packets or packets that come from an address in the prefix 10.108/16, and send an administratively-prohibited ICMP message for all packets that do not match:

```

[edit]
firewall {
    family inet {
        filter ospf-or-131 {
            term protocol-match {
                from {
                    protocol ospf;
                }
            }
        }
    }
}

```

```

    }
    term address-match {
        from {
            source-address {
                10.108.0.0/16;
            }
        }
    }
}
}
}
}
}

```

Example: Counting Both Accepted and Rejected Packets

Reject all addresses except 192.168.5.0/24. In the first term, the statement 192.168.5.2/24 **except** causes this address to be considered a mismatch and this address is passed to the next term in the filter. The address 0.0.0.0/0 in the first term matches all other packets, and these are counted, logged, and rejected. In the second term, all packets that passed through the first term (that is, packets whose address matches 192.168.5.2/24) are counted, logged, and accepted.

```

[edit]
firewall {
    family inet {
        filter fire1 {
            term 1 {
                from {
                    address {
                        192.168.5.0/24 except;
                        0.0.0.0/0;
                    }
                }
                then {
                    count reject-pref1-1;
                    log;
                    reject;
                }
            }
            term 2 {
                then {
                    count reject-pref1-2;
                    log;
                    accept;
                }
            }
        }
    }
}
}
}
}
}

```

Example: Blocking TCP Connections to a Certain Port Except from BGP Peers

Block all TCP connection attempts to port 179 from all requesters except the specified BGP peers:

```
[edit]
firewall {
  family inet {
    filter bgp179 {
      term 1 {
        from {
          source-address {
            0.0.0.0/0;
          }
          source-prefix-list {
            bgp179 except;
          }
          destination-port bgp;
        }
        then {
          reject;
        }
      }
      term 2 {
        then {
          accept;
        }
      }
    }
  }
}
```

Expand the prefix list `bgp179` to include all BGP group neighbors:

```
[edit policy-options]
prefix-list bgp179 {
  apply-path "protocols bgp group <*> neighbor <*>";
}
```

Apply the filter `bgp179` to interface `lo0`:

```
[edit interfaces lo0]
unit 0 {
  family inet {
    filter {
      input bgp179;
    }
    address 10.0.0.1/32;
  }
}
```

Example: Accepting Packets with Specific IPv6 TCP Flags

Configure a filter to match on IPv6 TCP flags:

```
[edit]
firewall {
  family inet6 {
    filter tcpfilt {
      term 1 {
```

```

        from {
            next-header tcp;
            tcp-flags syn;
        }
        then {
            count tcp_syn_pkt;
            log;
            accept;
        }
    }
}
}
}

```

Example: Setting a Rate Limit for Incoming Layer 2 Control Packets

Configure rate limiting for incoming Layer 2 control packets. In order to meet this requirement, you must configure an input filter with the family type `any` and apply this filter to the interface:

```

[edit]
firewall {
    policer p1 {
        if-exceeding {
            bandwidth-limit 5m;
            burst-size-limit 10m;
        }
        then discard;
    }
    policer p2 {
        if-exceeding {
            bandwidth-limit 40m;
            burst-size-limit 100m;
        }
        then discard;
    }
    policer p3 {
        if-exceeding {
            bandwidth-limit 600m;
            burst-size-limit 1g;
        }
        then discard;
    }
    interface-set ifset {
        fe-*;
    }
    family any {
        filter L2-filter {
            term t1 {
                from {
                    interface fe-0/0/0.0;
                }
                then policer p1;
            }
            term t2 {

```

```

        from {
            interface-set ifset;
        }
        then policer p2;
    }
    term t3 {
        then policer p3;
    }
}
}
}
[edit]
interfaces {
    fe-0/0/0 {
        unit 0 {
            family inet {
                address 10.1.1.1/30;
            }
        }
    }
    fe-1/0/0 {
        unit 0 {
            family inet {
                address 10.2.2.1/30;
            }
        }
    }
    lo0 {
        unit 0 {
            family any {
                filter {
                    input L2-filter;
                }
            }
        }
    }
}
}

```

Configuring Service Filters

A service filter identifies packets on one or more services are to be applied, and which PIC performs the service. To configure service filters, include the **service-filter** statement at the `[edit firewall family (inet | inet6)]` hierarchy level:

```

[edit firewall family (inet | inet6)]
service-filter filter-name {
    term term-name {
        from {
            match-conditions;
        }
        then {
            action;
            action-modifiers;
        }
    }
}

```

}



NOTE: You must specify either `inet` or `inet6` as the protocol family in order to configure a service filter.

Service filters are configured the same way as firewall filters. A subset of match conditions and actions for firewall filters are supported for service filters have a subset of the.

One of the actions you configure must be **service** or **skip**:

- Specifying the **service** action directs packets for stateful-firewall service.
- Specifying the **skip** action let packets bypass stateful-firewall service.

The following actions are also supported for service filters:

- **count** *counter-name*—Count the packet in the specified counter.
- **log**—Log the packet header information in a buffer within the Packet Forwarding Engine. You can access this information by issuing the **show firewall log** command.
- **port-mirror**—Port Mirror the packets
- **sample**—Sample the packets

For more information about services and service interfaces, see the *JUNOS Services Interfaces Configuration Guide*.

Configuring Simple Filters

Simple filters are recommended for metropolitan Ethernet applications. They are supported on Gigabit Ethernet intelligent queuing (IQ2) and Enhanced Queuing Dense Port Concentrator (EQ DPC) interfaces only. Unlike normal filters, simple filters are for IPv4 traffic only and have the following restrictions:

- The **next-term** action is not supported.
- Qualifiers, such as **except** and **protocol-except** match conditions, are not supported.
- Noncontiguous masks are not supported.
- Only one **source-address** and one **destination-address** prefix are allowed for each filter term. If you configure, multiple prefixes, only the last one is used.
- Ranges are only valid as source or destination ports. For example, you can configure **source-port 400-500** or **destination-port 600-700**.
- Output filters are not supported. You can apply a simple filter to ingress traffic only.
- Simple filters are not supported for interfaces in an aggregated-Ethernet bundle.

- Explicitly configurable terminating actions, such as **accept**, **reject**, or **discard**, are not supported. Simple filters always accept packets.
- Simple filters support only the following action modifiers: **forwarding-class**, **loss-priority**, and **policer**.

To configure simple filters, include the **simple-filter** statement at the [edit firewall family inet] hierarchy level:

```
[edit firewall family inet]
simple-filter filter-name {
  term term-name {
    from {
      match-conditions;
    }
    then {
      action-modifiers;
    }
  }
}
```

For more information about Ethernet IQ2 PICs and EQ DPCs and related features, see the *JUNOS Services Interfaces Configuration Guide* and the *JUNOS Class of Service Configuration Guide*. For additional information about configuring the MX Series routers, on which EQ DPCs are supported, see the *JUNOS MX Series Ethernet Services Routers Layer 2 Configuration Guide*.

Example: Configuring a Simple Filter

Configure a simple filter to support Ethernet IQ2 PICs:

```
[edit]
firewall {
  family inet {
    simple-filter sf-1 {
      term 1 {
        from {
          source-address 172.16.0.0/16;
          destination-address 20.16.0.0/16;
          source-port 1024-9071;
        }
        then {
          forwarding-class fc-be1;
          loss-priority high;
          accept;
        }
      }
      term 2 {
        from {
          source-address 173.16.0.0/16;
          destination-address 21.16.0.0/16;
        }
        then {
          forwarding-class fc-ef1;
          loss-priority low;
        }
      }
    }
  }
}
```

```

        accept;
    }
}
}
}
}

```

Configuring Firewall Filters for Logical Systems

You can configure a separate set of firewall filters for each logical system on the router. To configure a firewall filter for a logical system, you must perform at least the following tasks:

- Configure firewall filters for the logical system—To configure firewall filters for the logical system, include the `firewall` statement at the `[edit logical-systems logical-system-name]` hierarchy level:

```

[edit logical-systems logical-system-name]
firewall {
  family family-name {
    filter filter-name {
      accounting-profile name;
      interface-specific;
      term term-name {
        from {
          match-conditions;
        }
        then {
          action;
          action-modifiers;
        }
      }
    }
  }
}

```

- Apply firewall filters to interfaces in the logical system—To have the firewall filter take effect, you must apply it to an interface in the logical system by including the `filter` statement at the `[edit logical-systems logical-system-name interfaces interface-name unit logical-unit-number family family-name]` hierarchy level:

```

[edit logical-systems logical-system-name interfaces interface-name unit
  logical-unit-number family family-name]
filter {
  input filter-name;
  output filter-name;
}

```

To identify firewall objects configured under logical systems, operational `show` commands and firewall-related SNMP MIB objects include a `__logical-system-name/` prefix in the object name. For example, firewall objects configured under the `ls1` logical system include an `__ls1/` prefix.

This section includes the following topics:

- Guidelines for Firewall Configuration in Logical Systems on page 235
- Unsupported Configuration Statements, Actions, and Action Modifiers on page 242

Guidelines for Firewall Configuration in Logical Systems

As a general rule, firewall filters configured under a logical system must be complete and self-contained. Typically, the filters cannot reference firewall elements configured at the `[edit firewall]` hierarchy level or at another `[edit logical-systems logical-system-name]` hierarchy level. If no firewall filters are configured for a logical system, the firewall filters at the `[edit firewall]` hierarchy level are applied.

In some situations, `firewall` statements that are valid under the `[edit firewall]` hierarchy are not supported under the `[edit logical-systems logical-system-name firewall]` hierarchy. There are three scenarios to consider:

- Scenario 1. An object in the firewall hierarchy references another object in the hierarchy; for example, when a firewall filter references a firewall policer.
- Scenario 2. An object outside the firewall references an object inside the firewall hierarchy; for example, a firewall filter is applied to an interface.
- Scenario 3. An object in the firewall hierarchy references an object outside the firewall hierarchy; for example, when a firewall filter references a prefix list (defined under the `[edit policy-options]` hierarchy).

This section includes the following topics:

- Scenario 1: Firewall Objects Reference Other Firewall Objects on page 235
- Scenario 2: Nonfirewall Objects Reference Firewall Objects on page 236
- Scenario 3: Firewall Objects Reference Nonfirewall Objects on page 240

Scenario 1: Firewall Objects Reference Other Firewall Objects

If a firewall object references a subordinate object (for example, a policer or prefix list), that subordinate object must be defined within the firewall object. For example, if a firewall filter configuration references a policer, that policer must be configured under the same firewall object as the filter. This rule applies even if the same policer is configured under the main firewall configuration or if the same policer is configured as part of a firewall in another logical system.

In this example, the `filter1` filter references the `pol1` policer. Both `filter1` and `pol1` are defined under the same firewall object. This configuration is valid. If `pol1` were defined under another firewall object, the configuration would not be valid.

```
[edit]
logical systems {
  ls1 {
    firewall {
      policer pol1 {
        if-exceeding {
          bandwidth-limit 401k;
```

```

        burst-size-limit 50k;
    }
    then discard;
}
filter filter1 {
    term one {
        from {
            source-address 12.1.0.0/16;
        }
        then {
            reject host-unknown;
        }
    }
    term two {
        from {
            source-address 12.2.0.0/16;
        }
        then policer pol1;
    }
}
}
}
}

```

Scenario 2: Nonfirewall Objects Reference Firewall Objects

When an object is configured within a logical system (but is not included in the firewall configuration for the logical system) and that object references a firewall object, the following logic is used to resolve the configuration:

- If firewall configuration statements are defined within the same logical system, the [edit logical-systems *logical-system-name* firewall] hierarchy is searched to resolve the configuration. The main [edit firewall] hierarchy is *not* searched.
- If no firewall configuration statements are defined within the same logical system, the firewall configuration defined at the [edit firewall] hierarchy level is searched to resolve the configuration. This search option is provided for legacy purposes. The main [edit firewall] hierarchy is searched only if firewall configuration statements are *not* defined within the same logical system.
- Firewall configurations that belong to other logical systems are *not* searched.

In the following example, the filter **fred** is applied to an interface in the logical system **ls1**. However, **fred** is defined in the main firewall configuration instead of in the **ls1** firewall configuration. Therefore, in this first example, the configuration is not valid.

```

[edit]
logical-systems {
    ls1 {
        interfaces {
            fe-0/3/2 {
                unit 0 {
                    family inet {
                        filter {
                            input-list [ filter1 fred ];
                        }
                    }
                }
            }
        }
    }
}

```

```

    }
  }
}
}
firewall {
  policer pol1 {
    if-exceeding {
      bandwidth-limit 401k;
      burst-size-limit 50k;
    }
    then discard;
  }
  filter filter1 {
    term one {
      from {
        source-address 12.1.0.0/16;
      }
      then {
        reject host-unknown;
      }
    }
    term two {
      from {
        source-address 12.2.0.0/16;
      }
      then policer pol1;
    }
  }
}
}
}
firewall {
  policer pol1 {
    if-exceeding {
      bandwidth-limit 701k;
      burst-size-limit 70k;
    }
    then discard;
  }
  family inet {
    filter fred {
      term one {
        from {
          source-address 11.1.0.0/16;
        }
        then {
          log;
          reject host-unknown;
        }
      }
    }
  }
}
}

```

To fix this example, define filter **fred** under logical system **ls1**. In this case, the filter **fred** applied to interface **fe-0/3/2** looks for source address **10.1.0.0/16** rather than **11.1.0.0/16**.

```
[edit]
logical-systems {
  ls1 {
    interfaces {
      fe-0/3/2 {
        unit 0 {
          family inet {
            filter {
              input-list [ filter1 fred ];
            }
          }
        }
      }
    }
  }
  firewall {
    policer pol1 {
      if-exceeding {
        bandwidth-limit 401k;
        burst-size-limit 50k;
      }
      then discard;
    }
    filter filter1 {
      term one {
        from {
          source-address 12.1.0.0/16;
        }
        then {
          reject host-unknown;
        }
      }
      term two {
        from {
          source-address 12.2.0.0/16;
        }
        then policer pol1;
      }
    }
    family inet {
      filter fred {
        term one {
          from {
            source-address 10.1.0.0/16;
          }
          then {
            log;
            reject host-unknown;
          }
        }
      }
    }
  }
}
```

```

    }
  }
  firewall {
    policer pol1 {
      if-exceeding {
        bandwidth-limit 701k;
        burst-size-limit 70k;
      }
      then discard;
    }
    family inet {
      filter fred {
        term one {
          from {
            source-address 11.1.0.0/16;
          }
          then {
            log;
            reject host-unknown;
          }
        }
      }
    }
  }
}

```

If, however, the [edit logical-systems logical-system-name] hierarchy does not contain any `firewall` statements, then the main firewall configuration is used for any filter or policer references. For example, the following configuration is also allowed:

```

[edit]
logical-systems {
  ls1 {
    interfaces {
      fe-0/3/2 {
        unit 0 {
          family inet {
            filter {
              input-list [ filter1 fred ];
            }
          }
        }
      }
    }
  }
}
firewall {
  policer pol1 {
    if-exceeding {
      bandwidth-limit 701k;
      burst-size-limit 70k;
    }
    then discard;
  }
  family inet {
    filter fred {
      term one {

```

```

        from {
            source-address 11.1.0.0/16;
        }
        then {
            log;
            reject host-unknown;
        }
    }
}
filter filter1 {
    term one {
        from {
            source-address 12.1.0.0/16;
        }
        then {
            reject host-unknown;
        }
    }
    term two {
        from {
            source-address 12.2.0.0/16;
        }
        then policer pol1;
    }
}
}
}

```

Scenario 3: Firewall Objects Reference Nonfirewall Objects

In many cases, a firewall configuration references objects outside the firewall configuration. As a general rule, the referenced object must be defined under the same logical system as the referencing object. However, there are cases when the configuration of the referenced object is not supported at the `[edit logical-systems logical-system-name]` hierarchy level.

In the following example, the service filter `inetsf1` references prefix list `prefix1`. The service set `fred` cannot be defined under the logical system `lr1`. In this case, the `[edit services]` hierarchy is searched for the definition of the `fred` service set. This configuration is allowed because the `[edit logical-systems logical-system logical-system-name]` hierarchy already had the capability to reference service sets outside the logical system hierarchy.

```

[edit]
logical-systems {
    ls1 {
        interfaces {
            fe-0/3/2 {
                unit 0 {
                    family inet {
                        service {
                            input {
                                service-set fred service-filter lr1inetsf1;
                            }
                        }
                    }
                }
            }
        }
    }
}

```



```

    }
  }
}
policy-options {
  prefix-list prefix1 {
    1.1.0.0/16;
    1.2.0.0/16;
    1.3.0.0/16;
  }
}
firewall {
  policer pol1 {
    if-exceeding {
      bandwidth-limit 401k;
      burst-size-limit 50k;
    }
    then discard;
  }
  filter filter1 {
    term one {
      from {
        source-address 12.1.0.0/16;
      }
      then {
        reject host-unknown;
      }
    }
    term two {
      from {
        source-address 12.2.0.0/16;
      }
      then policer pol1;
    }
  }
  family inet {
    service-filter inetsf1 {
      term term1 {
        from {
          source-prefix-list {
            prefix1;
          }
        }
        then count prefix1;
      }
    }
  }
}
}
}
}
services {
  service-set fred {
    max-flows 100;
    interface-service {
      service-interface sp-1/2/0.0;
    }
  }
}

```

```

    }
}

```

Unsupported Configuration Statements, Actions, and Action Modifiers

Table 33 on page 242 lists statements that are supported at the [edit firewall] hierarchy level but not at the [edit logical-systems *logical-system-name* firewall] hierarchy level.

Table 33: Unsupported Firewall Statements for Logical Systems

Statement	Example	Description
accounting-profile	<pre> [edit] logical-systems { ls1 { firewall { family inet { filter myfilter { accounting-profile fw-profile; ... } } } } } </pre>	<p>In this example, the <code>accounting-profile</code> statement is not allowed because the accounting profile <code>fw-profile</code> is configured under the [edit <code>accounting-options</code>] hierarchy.</p>
load-balance-group	<pre> [edit] logical-systems { ls1 { firewall { load-balance-group lb-group { next-hop-group nh-group; } } } } </pre>	<p>This configuration is not allowed because the <code>next-hop-group nh-group</code> statement must be configured at the [edit <code>forwarding-options next-hop-group</code>] hierarchy level—outside the [edit <code>logical-systems logical-system-name firewall</code>] hierarchy.</p> <p>Currently, the <code>forwarding-options dhcp-relay</code> statement is the only forwarding option supported for logical systems.</p>

Table 33: Unsupported Firewall Statements for Logical Systems (*continued*)

Statement	Example	Description
virtual-channel	<pre>[edit] logical-systems { ls1 { firewall { family inet { filter foo { term one { from { source-address 10.1.0.0/16; } then { virtual-channel sammy; } } } } } } }</pre>	This configuration is not allowed because the virtual channel sammy refers to an object defined at the [edit class-of-service] hierarchy level and class of service is not supported for logical systems.

Table 34 on page 243 includes a list of the firewall filter actions and action modifiers that are supported at the **[edit firewall]** hierarchy level, but not supported at the **[edit logical-systems logical-system-name firewall]** hierarchy level.

Table 34: Unsupported Firewall Actions and Action Modifiers for Logical Systems

Action or Action Modifier	Example	Description
analyzer	<pre>[edit] logical-systems { ls1 { firewall { family inet { filter foo { term one { from { source-address 10.1.0.0/16; } then { analyzer; } } } } } } }</pre>	(EX Series switches) Because the analyzer action relies on a configuration defined at the [edit ethernet-switching-options] hierarchy level, this action is not supported.

Table 34: Unsupported Firewall Actions and Action Modifiers for Logical Systems *(continued)*

Action or Action Modifier	Example	Description
ipsec-sa	<pre> [edit] logical-systems { ls1 { firewall { family inet { filter foo { term one { from { source-address 10.1.0.0/16; } then { ipsec-sa barney; } } } } } } } </pre>	Because the ipsec-sa action modifier references barney , a security association defined outside the local logical system, this action is not supported.
logical-system	<pre> [edit] logical-systems { ls1 { firewall { family inet { filter foo { term one { from { source-address 10.1.0.0/16; } then { logical-system fred; } } } } } } } </pre>	Because the logical-system action refers to fred , a logical system defined outside the local logical system, this action is not supported.

Table 34: Unsupported Firewall Actions and Action Modifiers for Logical Systems *(continued)*

Action or Action Modifier	Example	Description
next-hop-group	<pre>[edit] logical-systems { ls1 { firewall { family inet { filter foo { term one { from { source-address 10.1.0.0/16; } then { next-hop-group fred; } } } } } } }</pre>	Because the <code>next-hop-group</code> action refers to <code>fred</code> , an object defined at the <code>[edit forwarding-options next-hop-group]</code> hierarchy level, this action is not supported.
port-mirror	<pre>[edit] logical-systems { ls1 { firewall { family inet { filter foo { term one { from { source-address 10.1.0.0/16; } then { port-mirror; } } } } } } }</pre>	Because the <code>port-mirror</code> action relies on a configuration defined at the <code>[edit forwarding-options port-mirroring]</code> hierarchy level, this action is not supported.

Table 34: Unsupported Firewall Actions and Action Modifiers for Logical Systems (*continued*)

Action or Action Modifier	Example	Description
sample	<pre> [edit] logical-systems { ls1 { firewall { family inet { filter foo { term one { from { source-address 10.1.0.0/16; } then { sample; } } } } } } } </pre>	<p>In this example, the sample action depends on the sampling configuration defined under the [edit forwarding-options] hierarchy. Therefore, the sample action is not supported.</p>
syslog	<pre> [edit] logical-systems { ls1 { firewall { family inet { filter icmp-syslog { term icmp-match { from { address { 192.168.207.222/32; } protocol icmp; } then { count packets; syslog; accept; } } term default { then accept; } } } } } } </pre>	<p>In this example, there must be at least one system log (system syslog file filename) with the firewall facility enabled for the icmp-syslog filter's logs to be stored.</p> <p>Because this firewall configuration relies on a configuration outside the logical system, the syslog action modifier is not supported.</p>

Configuring Accounting for Firewall Filters

Juniper Networks devices can collect various kinds of data about traffic passing through the device. You can set up one or more *accounting profiles* that specify some common characteristics of this data, including the following:

- Fields used in the accounting records
- Number of files that the routing platform retains before discarding, and the number of bytes per file
- Polling period that the system uses to record the data

There are several types of accounting profiles: interface, firewall filter, destination class, and Routing Engine. To configure an accounting profile, include statements at the [edit accounting-options] hierarchy level. For more information, see the *JUNOS Network Management Configuration Guide*.

To activate a firewall filter profile, include the `accounting-profile` statement at the [edit firewall family *family-name* filter *filter-name*] hierarchy level:

```
[edit firewall family family-name filter filter-name]
  accounting-profile profile-name;
```

If you apply the same profile name to both a firewall filter and an interface, it causes an error.

The following example configures an accounting profile called `fw_profile` and applies it to the firewall filter called `myfilter`.

```
[edit]
accounting-options {
  filter-profile fw_profile {
    file fw_accounting;
    interval 60;
    counters {
      counter1;
      counter2;
      counter3;
    }
  }
}
firewall {
  family inet {
    filter myfilter {
      accounting-profile fw_profile;
      ...
      term accept-all {
        then {
          count counter1;
          accept;
        }
      }
    }
  }
}
```

}

Configuring Filter-Based Forwarding

You can configure filters to classify packets based on source address and specify the forwarding path the packets take within the router by configuring a filter on the ingress interface. For example, you can use this filter for applications to differentiate traffic from two clients that have a common access layer (for example, a Layer 2 switch) but are connected to different Internet service providers (ISPs). When the filter is applied, the router can differentiate the two traffic streams and direct each to the appropriate network. Depending on the media type the client is using, the filter can use the source IP address to forward the traffic to the corresponding network through a tunnel. You can also configure filters to classify packets based on IP protocol type or IP precedence bits.



NOTE: Source-class usage filter matching and unicast reverse-path forwarding checks are not supported on an interface configured with filter-based forwarding (FBF).

You can also forward packets based on output filters by configuring a filter on the egress interfaces. In the case of port mirroring, it is useful for port-mirrored packets to be distributed to multiple monitoring PICs and collection PICs based on patterns in packet headers. FBF on the port-mirroring egress interface must be configured.

Packets forwarded to the output filter have been through at least one route lookup when an FBF filter is configured on the egress interface. After the packet is classified at the egress interface by the FBF filter, it is redirected to another routing table for further route lookup.

Filter-based forwarding is supported for IPv4 and IPv6.

To direct traffic meeting defined match conditions to a specific routing instance, include the **routing-instance** filter action:

```
routing-instance routing-instance;
```

For IPv4 traffic, include the action at the [edit firewall family inet filter *filter-name* term *term-name* then] hierarchy level. For IPv6 traffic, include the action at the [edit firewall family inet6 filter *filter-name* term *term-name* then] hierarchy level. For MPLS traffic, configure the filter terms at the [edit firewall family mpls filter *filter-name* term *term-name* then] hierarchy level.

The **routing-instance** filter action accepts the traffic meeting the match conditions and directs it to the routing instance named in *routing-instance*. For information about forwarding instances and routing instances, see the *JUNOS Routing Protocols Configuration Guide*.



NOTE: In JUNOS Release 9.0 and later, you can no longer specify a routing-instance name of *default* or include special characters within the name of a routing instance.

To complete the configuration, you must also create a routing table group that adds interface routes to the following routing instances:

- Routing instance named in the action
- Default routing table `inet.0`

You create a routing table group to resolve the routes installed in the routing instance to directly connected next hops on that interface. For more information on routing table groups and interface routes, see the *JUNOS Routing Protocols Configuration Guide*.

Examples: Configuring Filter-Based Forwarding

Configure a filter to direct traffic to ISP1 or ISP2 based on source address matching:

```
[edit firewall]
family inet {
  filter classify-customers {
    term isp1-customers {
      from {
        source-address 10.1.1.0/24;
        source-address 10.1.2.0/24;
      }
      then {
        routing-instance isp1-route-table;
      }
    }
    term isp2-customers {
      from {
        source-address 10.2.1.0/24;
        source-address 10.2.2.0/24;
      }
      then {
        routing-instance isp2-route-table;
      }
    }
    term default {
      then {
        accept;
      }
    }
  }
}
```

Configure a filter-based forwarding (FBF) filter for family `inet6`:

```
[edit]
firewall {
  family inet6 {
    filter ftf_fbf {
      term 0 {
        from {
          source-address {
            ::10.34.1.0/120;
          }
        }
      }
    }
  }
}
```

```

    }
  }
  then {
    count ce1;
    log;
    routing-instance ce1;
  }
}
term 1 {
  from {
    source-address {
      ::10.34.2.0/120;
    }
  }
  then {
    count ce2;
    log;
    routing-instance ce2;
  }
}
term default {
  then {
    count default;
    accept;
  }
}
}
}
}

```

Configuring Forwarding Table Filters

The following sections describe the following topics:

- Overview of Forwarding Table Filters on page 250
- Configuring a Forwarding Table Filter on page 251

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.

For more detailed information about configuring filters, see *Configuring Standard Firewall Filters*.

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-instances 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;
}
```

For more information about applying forwarding table filters, see “Applying Filters to Forwarding Tables” on page 329. For information about routing instances, see the *JUNOS Routing Protocols Configuration Guide*.

Configuring System Logging of Firewall Filter Operations

System logging can be configured for the firewall filter process. You can set system logging to record messages of a particular level or all levels. The messages are sent to a system logging file.

The following is a sample system logging configuration for the firewall filter `icmp-syslog`. For more information about configuring system logging, see the *JUNOS System Basics Configuration Guide*.

```
[edit]
system {
  syslog {
    file filter {
      firewall any;
      archive no-world-readable;
    }
  }
}
```

This configuration causes the system log to write any messages with the `syslog` facility of `firewall` to the file `/var/log/filter`. This keeps the messages out of the main system log file and makes them easier to find.

Example: Configuring Firewall Filter System Logging

Create a filter that logs and counts ICMP packets that have `192.168.207.222` as either their source or destination:

```
[edit]
firewall {
  family inet {
    filter icmp-syslog {
      term icmp-match {
        from {
          address {
            192.168.207.222/32;
          }
          protocol icmp;
        }
        then {
          count packets;
          syslog;
          accept;
        }
      }
      term default {
        then accept;
      }
    }
  }
}
```

Enter the `show log filter` command to display the results:

```
root@hostname> show log filter
Mar 20 08:03:11 hostname feb FW: so-0/1/0.0   A icmp 192.168.207.222
192.168.207.223      0      0 (1 packets)
```

This output file contains the following fields:

- **Date and Time**—Date and time at which the packet was received (not shown in the default).
- **Filter action:**
 - **A**—Accept (or next term)
 - **D**—Discard
 - **R**—Reject
- **Protocol**—Packet's protocol name or number.
- **Source address**—Source IP address in the packet.
- **Destination address**—Destination IP address in the packet.



NOTE: If the protocol is ICMP, the ICMP type and code are displayed. For all other protocols, the source and destination ports are displayed.

The last two fields (both zero) are the source and destination TCP/UDP ports, respectively, and are shown for TCP or UDP packets only. This log message indicates that only one packet for this match has been detected in about a one-second interval. If packets arrive faster, the system log function compresses the information so that less output is generated, and displays an output similar to the following:

```
root@hostname> show log filter
Mar 20 08:08:45 hostname feb FW: so-0/1/0.0  A icmp 192.168.207.222
192.168.207.223      0      0 (515 packets)
```

Chapter 10

Policer Overview

Policing, or rate limiting, enables you to limit the amount of traffic that passes into or out of an interface. It is an essential component of firewall filters that is designed to thwart denial-of-service (DoS) attacks. Policing applies two types of rate limits on the traffic:

- Bandwidth—The number of bits per second permitted, on average.
- Maximum burst size—The maximum size permitted for bursts of data that exceed the given bandwidth limit.

Policing uses the *token-bucket algorithm*, which enforces a limit on average bandwidth while allowing bursts up to a specified maximum value. It offers more flexibility than the *leaky bucket algorithm* (see the *JUNOS Class of Service Configuration Guide*) in allowing a certain amount of bursty traffic before it starts discarding packets.

You can define specific classes of traffic on an interface and apply a set of rate limits to each. You can use a policer in one of two ways: as part of a filter configuration or as part of a logical interface (where the policer is applied to all traffic on that interface).

After you have defined and named a policer, it is stored as a template. You can later use the same policer name to provide the same policer configuration each time you wish to use it. This eliminates the need to define the same policer values more than once.

Chapter 11

Policer Configuration

The following sections describe the tasks required for configuring policers and provide configuration examples:

- Configuring Policers on page 257
- Minimum Policer Configuration on page 259
- Configuring Policers on page 259
- Configuring Multifield Classifiers for Policing on page 262
- Configuring Interface Sets on page 269
- Applying Interface Policers on page 270
- Configuring Aggregate Policers on page 271
- Configuring a Hierarchical Policer on page 272
- Physical Interface Policer Overview on page 273
- Configuring Physical Interface Policers on page 273
- Configuring Bandwidth Policers on page 277
- Configuring Load-Balance Groups on page 278
- Configuring Tricolor Marking on page 278
- Examples: Configuring Policing on page 282

Configuring Policers

To configure policers, you include statements at the [edit firewall] hierarchy level of the configuration:

```
[edit firewall]
policer policer-name {
  filter-specific;
  if-exceeding {
    bandwidth-limit bps;
    bandwidth-percent number;
    burst-size-limit bytes;
  }
  logical-bandwidth-policer;
  logical-interface-policer;
  physical-interface-policer;
  then {
    policer-action;
  }
}
```

```

    }
  }
  [Unresolved xref] hierarchical-policer-name {
    aggregate {
      if-exceeding {
        bandwidth-limit bps;
        burst-size-limit bytes;
      }
      then {
        policer-action;
      }
    }
    premium {
      if-exceeding {
        bandwidth-limit bps;
        burst-size-limit bytes;
      }
      then {
        policer-action;
      }
    }
  }
}
interface-set interface-set-name {
  interface-name;
}
family family-name {
  filter filter-name {
    accounting-profile name;
    interface-specific;
  }
  prefix-action name {
    count;
    destination-prefix-length prefix-length;
    policer policer-name;
    source-prefix-length prefix-length;
    subnet-prefix-length prefix-length;
  }
}
load-balance-group group-name {
  next-hop-group [ group-names ];
}
three-color-policer name {
  action {
    loss-priority high then discard;
  }
  logical-interface-policer;
  single-rate {
    (color-aware | color-blind);
    committed-information-rate bps;
    committed-burst-size bytes;
    excess-burst-size bytes;
  }
  two-rate {
    (color-aware | color-blind);
    committed-information-rate bps;
    committed-burst-size bytes;
  }
}

```

```

        peak-information-rate bps;
        peak-burst-size bytes;
    }
}

```

Minimum Policer Configuration

To configure a policer, you must perform at least the following tasks:

- Configure policers—To configure policers, include the **policer** statement at the [edit firewall] hierarchy level. After policers are defined, you reference them in the **then** clause of a term:

```

[edit firewall]
policer policer-name {
  if-exceeding {
    bandwidth-limit bps;
    bandwidth-percent number;
    burst-size-limit bytes;
  }
  then {
    policer-action;
  }
}
family family-name {
  filter filter-name {
  }
}

```

- Add actions, such as **accept**, **discard**, or **next term**, or action modifiers, such as **count** or **log**.
- Apply the policers to an interface to activate them.

The policer is applied to the packet first, and if the packet exceeds the defined limits, the actions of the **then** clause of the policer are applied. If the result of the policing action is not a discard, the remaining components of the **then** clause of the term are applied.



NOTE: If an input filter is configured on the same logical interface as the policer, the policer is executed first.

To display statistics about a filter statement policer configuration, use the **show policers** command.

Configuring Policers

You can configure a new policer for each filter or term that requires policing. To configure term-specific policers, include the **policer** statement at the [edit firewall] hierarchy level:

```
[edit firewall]
policer policer-name {
  if-exceeding {
    bandwidth-limit bps;
    bandwidth-percent number;
    burst-size-limit bytes;
  }
  then {
  }
}
```

The following sections describe the components of the **policer** statement and provide policer configuration examples:

- Configuring Rate Limiting on page 260
- Configuring Policer Actions on page 261

Configuring Rate Limiting

To specify the rate limiting part of a policer, include an **if-exceeding** statement at the **[edit firewall policer *policer-name*]** hierarchy level:

```
[edit firewall policer]
if-exceeding {
  bandwidth-limit bps;
  bandwidth-percent number;
  burst-size-limit bytes;
}
```

You specify the bandwidth limit in bits per second (bps). You can specify the value as a complete decimal number or as a decimal number followed by the abbreviation k (1000), m (1,000,000), or g (1,000,000,000). Any value below 61,040 bps results in an effective rate of 30,520 bps. In JUNOS Release 9.4 and later, the minimum bandwidth limit that you can configure on M120, M320, and MX Series routers only is 8000 bps. The minimum bandwidth limit that you can configure for all other platforms remains 32,000 bps. The maximum bandwidth limit is 40 gigabits per second (Gbps).

You can rate-limit traffic based upon port speed. This port speed can be specified by a bandwidth percentage in a policer. You must specify the percentage as a complete decimal number between 1 and 100.



NOTE: You cannot rate-limit based on bandwidth percentage for aggregate, tunnel, and software interfaces. The bandwidth percentage policer cannot be used for forwarding table filters. Bandwidth percentage policers can only be used for interface-specific filters.

The maximum burst size controls the amount of traffic bursting allowed. To determine the value for the burst-size limit, the preferred method is to multiply the bandwidth (expressed as bytes per second) of the interface on which you are applying the filter by the amount of time you allow a burst of traffic at that bandwidth to occur. We

recommend that you use a value of 5 ms as the starting point for the allowable amount of time for a burst of traffic.

If you express the bandwidth as bits per second, use the following formula to calculate the burst size.

$$\text{burst size} = \text{bandwidth} \times \text{allowable time for burst traffic} / 8$$

If you do not know the interface bandwidth, you can multiply the maximum transmission unit (MTU) of the traffic on the interface by 10 to obtain a value. For example, the burst size for an MTU of 4700 would be 47,000 bytes. At minimum, burst size should be at least 10 interface MTUs. The maximum value for the burst-size limit is 100 megabits per second (Mbps).

For a sample filter configuration for rate limiting, see “Examples: Configuring Policing” on page 282.

Configuring Policer Actions

If a packet does not exceed its rate limits, it is processed further without being affected. If the packet exceeds its limits, it is handled in one of two ways, depending on what you specify:

- Discarded
- Marked for subsequent processing based on its loss priority and forwarding class

To configure a policer action, include the `then` statement at the `[edit firewall policer policer-name]` hierarchy level:

```
[edit firewall policer policer-name]
then {
}
```

Policer actions include one or more of the following:

- `discard`—Discard a packet that exceeds the rate limits.
- `loss-priority level`—Set the loss priority level to `low`, `medium-low`, `medium-high`, or `high`.
- `forwarding-class class-name`—Specify the forwarding class to any class name already configured for the forwarding class.

Example: Configuring a Policer Action

Discard any packet that exceeds a bandwidth of 300 kilobits per second (Kbps) and a burst-size limit of 500 kilobytes (KB):

```
[edit firewall]
policer p1 {
  if-exceeding {
    bandwidth-limit 300k;
    burst-size-limit 500k;
  }
}
```

```

    then {
        discard;
    }
}

```

Configuring Multifield Classifiers for Policing

Multifield classifiers take action on incoming or outgoing packets, depending whether the firewall rule is applied as an input filter or an output filter. When TCM is enabled, T Series and M320 routers support four multifield classifier packet loss priority (PLP) designations: **low**, **medium-low**, **medium-high**, and **high**.

To configure the PLP for a multifield classifier, include the **loss-priority** statement in a policer or firewall filter that you configure at the **[edit firewall]** hierarchy level:

```

[edit firewall]
family family-name {
  filter filter-name {
    term term-name {
      from {
        match-conditions;
      }
      then {
        loss-priority (low | medium-low | medium-high | high);
        forwarding-class class-name;
      }
    }
  }
}

```

The inputs (match conditions) for a multifield classifier are one or more of the six packet header fields: destination address, source address, IP protocol, source port, destination port, or DSCP. The outputs for a multifield classifier are the forwarding class, the PLP, or both. In other words, a multifield classifier sets the forwarding class and the PLP for each packet entering or exiting the interface with a specific destination address, source address, IP protocol, source port, destination port, or DSCP.

For example, in the following configuration, the forwarding class **expedited-forwarding** and PLP **medium-high** are assigned to all IPv4 packets with the **10.1.1.0/24** or **10.1.2.0/24** source address:

```

firewall {
  family inet {
    filter classify-customers {
      term isp1-customers {
        from {
          source-address 10.1.1.0/24;
          source-address 10.1.2.0/24;
        }
        then {
          loss-priority medium-high;
          forwarding-class expedited-forwarding;
        }
      }
    }
  }
}

```

```

    }
  }
}

```

To use this classifier, you must configure the settings for the `expedited-forwarding` forwarding class at the `[edit class-of-service forwarding-classes queue queue-number expedited-forwarding]` hierarchy level.



NOTE: Because the policer is executed before the filter, if an input policer is also configured on the logical interface, it cannot use the forwarding class and PLP of a multifield classifier associated with the interface.

You can configure *multifield classifiers* within a firewall filter to set the packet's forwarding class and packet loss priority. You can also apply policers to packets matching some classification term. The policing action might affect the resulting forwarding class, packet loss priority, and accept or drop status. For more information, see the *JUNOS Class of Service Configuration Guide*.

To configure the forwarding class and loss priority, include the `then` statement:

```

then {
  loss-priority;
  forwarding-class class-name;
}

```

You can include the statement at the following hierarchy levels:

- `[edit firewall filter filter-name term term-name]`
- `[edit firewall policer policer-name]`

You can specify one or both of the following actions:

- `loss-priority`—Set the loss priority level to `low` or `high`.
- `forwarding-class`—Specify the forwarding class to any class name already configured for the forwarding class.

For more information about forwarding class and loss priority, see the *JUNOS Class of Service Configuration Guide*. For more information about policers, see the following sections:

- Configuring Filter-Specific Policers on page 263
- Configuring Policer Actions for Specific Address Prefixes on page 264
- Examples: Classifying Traffic on page 268

Configuring Filter-Specific Policers

You can configure filter-specific policers within the firewall configuration. Filter-specific policers allow you to configure policers and counters for a specific filter name.

When you configure the **filter-specific** statement, a single policer set is created for the entire filter. All traffic matching the terms of the firewall filter with the action **policer** goes through that single policer. The default is a term-specific policer in which a single policer set is created for each term within the filter. All traffic matching the terms of the firewall filter with the action **policer** goes through the part of the policer that is specific to that term.

To configure filter-specific policers, include the **filter-specific** statement at the **[edit firewall policer *policer-name*]** hierarchy level:

```
[edit firewall policer policer-name]  
filter-specific;
```

If the **filter-specific** statement is not configured, then the policer defaults to a term-specific policer.

You can apply the filter-specific policers to the family **inet**.

Configuring Policer Actions for Specific Address Prefixes

You can configure prefix-specific actions within the firewall configuration. Prefix-specific actions allow you to configure policers and counters for specific addresses or ranges of addresses. This allows you to essentially create policers and counters on a per-prefix level.

To configure prefix-specific actions, include the **prefix-action *name*** statement at the **[edit firewall family *inet*]** hierarchy level:

```
[edit firewall family inet]  
prefix-action name {  
    count;  
    destination-prefix-length prefix-length;  
    policer policer-name;  
    source-prefix-length prefix-length;  
    subnet-prefix-length prefix-length;  
}
```

The following formula determines the number of prefix-specific actions created:

$$\text{Number} = 2^{\text{(source/destination-prefix-length - subnet-prefix-length)}}$$

The **subnet-prefix-length** statement allows for more control for the flexibility offered by prefix-specific actions, allowing the policers to be more applicable and powerful. For example, if you want to filter all Transmission Control Protocol (TCP) packets and define two policers, all packets ending with 0 in the last address bit increment the first policer, while all packets ending with 1 in the address bit increment the second policer. As another example, if you want to filter all TCP packets and define 256 policers, matching is based on the last octet of the destination address field. You achieve both cases by specifying an appropriate subnet prefix length.

Prefix-specific action is supported for the IP version 4 (IPv4) **inet** address family.

To configure prefix-specific actions, include the **prefix-action** statement and specify an action name.

To enable a prefix-specific counter, include the **count** statement.

To configure the destination address range specified for a prefix-specific policer or counter, include the **destination-prefix-length** statement.

To enable a set of prefix-specific policers, include the **policer** statement and specify the **policer** name.

To configure the source address range specified for a prefix-specific policer or counter, include the **source-prefix-length** statement.

To configure the total address range of the subnet supported, include the **subnet-prefix-length** statement. The source or destination prefix length must be larger than the subnet prefix length.

Prefix-specific action applies to a specific prefix length, and not to a specific interface. You can add an interface policer policies at the aggregate level for a specific interface. You could also use the **next term** action to configure all Hypertext Transfer Protocol (HTTP) traffic to each host to transmit at 500 Kbps and have the total HTTP traffic limited to 1 Mbps.

The maximum number of policers you can configure for one subnet is 65,536. If you configure more than 65,536 policers, you receive an error message.



NOTE: J Series Services Routers do not support prefix-specific actions.

Examples: Configuring Policer Actions for Specific Address Prefixes

Create a prefix-specific policer operating on the source address and apply it to the input interface:

```
[edit]
firewall {
  policer host-policer {
    filter-specific;
    if-exceeding {
      bandwidth-limit bps;
      burst-size-limit bytes;
    }
    then {
      discard;
    }
  }
}
family inet {
  prefix-action ftp-policer-set {
    count;
    destination-prefix-length 32;
    policer host-policer;
    subnet-prefix-length 24;
  }
  filter filter-ftp {
    term term1{
```

```

        from {
            destination-address 10.10.10/24;
            destination-port ftp;
        }
        then {
            prefix-action ftp-policer-set;
        }
    }
}
}
}

```

Filter all packets going to the /24 subnet, letting them pass to the prefix-specific action policers. In the policer set, the last octet of the source address field of the packet is used to index into the respective prefix-specific action policers.

```

[edit]
firewall {
    policer 1Mbps-policer {
        if-exceeding {
            bandwidth-limit 1m;
            burst-size-limit 63k;
        }
    }
    family inet {
        prefix-action per-source-policer {
            policer 1Mbps-policer;
            subnet-prefix-length 24;
            source-prefix-length 32;
        }
    }
    filter limit-all-hosts {
        term one {
            from {
                source-address {
                    10.10.10.0/24;
                }
            }
            then prefix-action per-source-policer;
        }
    }
}
}

```

In the preceding case, all packets are subjected to the prefix-specific action policing. The last octet of the source address field of the packet is used to index into the corresponding policer. In other words, all packets ending with 0x(xxx0000) match the first policer and all packets ending in 0x(xxx0001) match the second policer.

Therefore, 256 policers are created and shared by all addresses. In this case, 10.1.1.1, 10.2.2.1, 10.4.5.1 ... 10.x.x.1 share the same 1-Mbps policer; 10.1.1.2, 10.2.2.2, 10.4.5.2 ... 10.x.x.2 share another 1-Mbps policer, and so on.

Subject packets belonging to the 10.10.10.0/24 subnet are subject to policing by the prefix-specific action policers. Because 128 policers defined in the policer set, the /24 subnet can be thought of as being split into two /25 subnets, both of them

sharing the same prefix-specific action set. Therefore, 10.10.10.1 and 10.10.10.129 share the same 1-Mbps policer, 10.10.10.2 and 10.10.10.130 share another 1-Mbps policer, and so on.

```
[edit]
firewall {
  policer 1Mbps-policer {
    if-exceeding {
      bandwidth-limit 1m;
      burst-size-limit 63k;
    }
  }
  family inet {
    prefix-action per-source-policer {
      policer 1Mbps-policer;
      subnet-prefix-length 25;
      source-prefix-length 32;
    }
  }
  filter limit-all-hosts {
    term one {
      from {
        source-address {
          10.10.10.0/24;
        }
      }
      then prefix-action per-source-policer;
    }
  }
}
```

Define 256 policers based on the last octet of the source address field. However, you are only allowing a subset of that to pass through the match condition. As a result, only the lower half of the set is used.

```
[edit]
firewall {
  policer 1Mbps-policer {
    if-exceeding {
      bandwidth-limit 1m;
      burst-size-limit 63k;
    }
  }
  family inet {
    prefix-action per-source-policer {
      policer 1Mbps-policer;
      subnet-prefix-length 24;
      source-prefix-length 32;
    }
  }
  filter limit-all-hosts {
    term one {
      from {
        source-address {
          10.10.10.0/25;
        }
      }
    }
  }
}
```

```

    }
    then prefix-action per-source-policer;
  }
}

```

Accept packets from 10.10.10/24 and 10.11/16 subnets and subject them to policing by the same set of prefix-specific action policers. The policers are shared by packets across both subnets. There is a one-to-one correspondence between the 10.10.10/24 subnet. For 10.11/16, there is a many-to-one correspondence, as explained in the previous examples. Each of the 10.11.0/24, 10.11.1/24, 10.11.2/24 ... 10.11.255/24 subnets share the same prefix-specific action set.

Thus, 10.10.10.1, 10.11.1.1, 10.11.2.1 ... 10.11.x.1 share the same 1-Mbps policer; 10.10.10.2, 10.11.1.2, 10.11.2.2 ... 10.11.x.2 share another 1-Mbps policer, and so on.

```

[edit]
firewall {
  policer 1Mbps-policer {
    if-exceeding {
      bandwidth-limit 1m;
      burst-size-limit 63k;
    }
  }
  family inet {
    prefix-action per-source-policer {
      policer 1Mbps-policer;
      subnet-prefix-length 24;
      source-prefix-length 32;
    }
  }
  filter limit-all-hosts {
    term one {
      from {
        source-address {
          10.10.10/24;
          10.11/16;
        }
      }
      then prefix-action per-source-policer;
    }
  }
}

```

Examples: Classifying Traffic

Classify expedited forwarding traffic:

```

[edit]
firewall {
  policer ef-policer {
    if-exceeding {
      bandwidth-limit 300k;
      burst-size-limit 50k;
    }
  }
}

```

```

    }
    then {
        discard;
    }
}
term ef-multifield {
    then {
        loss-priority low;
        forwarding-class expedited-forwarding;
        policer ef-policer;
    }
}
}

```

Classify assured forwarding traffic:

```

firewall {
    policer af-policer {
        if-exceeding {
            bandwidth-limit 300k;
            burst-size-limit 500k;
        }
        then {
            loss-priority high;
        }
    }
    term af-multifield {
        then {
            loss-priority low;
            forwarding-class assured-forwarding;
            policer af-policer;
        }
    }
}

```

Configuring Interface Sets

In addition to including policers in firewall filters, you can configure an interface set that is not part of a firewall filter configuration. An interface set groups a number of interfaces into one interface set name.

To configure an interface set, include the `interface-set` statement at the `[edit firewall]` hierarchy level:

```

[edit firewall]
interface-set interface-set-name {
    interface-name;
}

```

You must specify more than one interface name to configure an interface set. This interface set can be used for firewall filter matching.

Applying Interface Policers

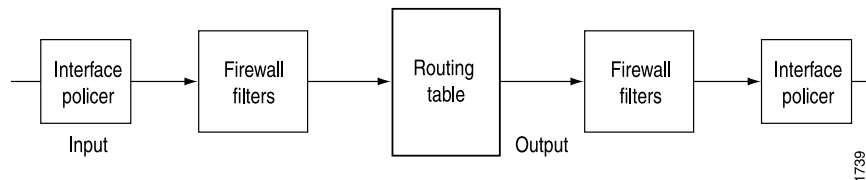
In addition to including policers in firewall filters, you can apply an interface policer that is not part of a firewall filter configuration. An interface policer can be applied to each family on an interface.

To apply an interface policer, include the `policer` statement at the `[edit interfaces interface-name unit logical-unit-number family family-name]` hierarchy level:

```
[edit interfaces interface-name unit logical-unit-number family family-name]
policer {
    input policer-name;
    output policer-name;
}
```

You must first configure the policer at the `[edit firewall]` hierarchy level before you can apply it to an interface. Both input and output policers are allowed, and can be used in conjunction with existing firewall filters. Input interface policers are evaluated before any input firewall filters. Likewise, output interface policers are evaluated after any output firewall filters (see Figure 12 on page 270).

Figure 12: Incoming and Outgoing Interface Policers



To display a policer on a particular interface, issue the `show interfaces policers` command at the command-line interface (CLI).



NOTE: This type of policer can only be applied to unicast packets. For information on configuring a filter for flooded traffic, see “Applying Filters to Forwarding Tables” on page 329.

Example: Applying an Interface Policer

Apply a policer on circuit cross-connect (CCC) interfaces:

```
[edit interfaces]
so-0/0/0 {
    encapsulation ppp-ccc;
    unit 0 {
        family ccc {
            policer {
                input dragnet;
            }
        }
    }
}
```

```
}
}
```

Configuring Aggregate Policers

You can configure a single aggregated policer to limit traffic on the same interface without the use of multiple instances of the same policer. Instead of policing each address family individually on an interface, you can aggregate policing with one policer. This single aggregated policer is also known as the logical interface policer.

To configure a logical interface policer, include the **logical-interface-policer** statement at the **[edit firewall policer *policer-name*]** hierarchy level:

```
logical-interface-policer;
```

You can configure rate limiting on the logical interface policer. For information on configuring rate limiting, see “Configuring Rate Limiting” on page 260. You can configure a policer action for the logical interface policer. For information on configuring policy actions, see “Configuring Policer Actions” on page 261.

After configuring the aggregated logical interface policer, you can apply the policer to an interface. To apply an aggregated logical interface policer, include the **policer *policer-name*** option at the **[edit interfaces *interface-name* unit 0 family *family-name*]** hierarchy level:

```
policer policer-name;
```

For more information about applying policers, see the *JUNOS Class of Service Configuration Guide*.

Example: Configuring an Aggregate Policer

Configure an aggregate policer to perform rating limiting:

```
[edit firewall policer new-police1]
if-exceeding {
  bandwidth-limit 100m;
  burst-size-limit 500k;
}
logical-interface-policer;
then {
  discard;
}
```

Apply the aggregate policer to rate-limit IPv4 and IPv6 traffic on interface **fe-0/1/1**:

```
[edit interfaces fe-0/1/1 unit 0 family inet]
policer new-police1;
[edit interfaces fe-0/1/1 unit 0 family inet6]
policer new-police1;
```

Configuring a Hierarchical Policier

The Enhanced IQ PIC can police traffic at Layer 2 in a hierarchical manner. Hierarchical policing maintains two rates: an aggregate rate and a high-priority rate. The traffic is marked differently depending on class of service, currently expedited forwarding and nonexpedited forwarding.

To configure a hierarchical policier, include the **aggregate** and **premium** statements at the [edit firewall hierarchical-policier *hierarchical-policier-name*] hierarchy level:

```
aggregate {
  if-exceeding {
    bandwidth-limit bps;
    burst-size-limit bytes;
  }
  then {
    policer-action;
  }
}
premium {
  if-exceeding {
    bandwidth-limit bps;
    burst-size-limit bytes;
  }
  then {
    policer-action;
  }
}
```

You can configure rate limiting on the hierarchical policier. For information about configuring rate limiting, see “Configuring Rate Limiting” on page 260. You can configure a policier action for the hierarchical policier. For information about configuring policy actions, see “Configuring Policier Actions” on page 261.

Use the **aggregate** statement to configure a rate limit and policy actions for all traffic levels (premium and normal). Use the **premium** statement to configure a rate limit and policy actions for premium traffic levels.

After configuring the hierarchical policier, you can apply the policier to the logical or physical interface on the Enhanced IQ PIC. To apply a hierarchical policier, include the policier *policier-name* option at the [edit interfaces *interface-name* unit 0 family *family-name*] hierarchy level:

```
policier policier-name;
```

For more information about applying hierarchical policiers, see the *JUNOS Class of Service Configuration Guide*.

Physical Interface Policer Overview

Physical interface policers permit you to configure a single aggregate policer that can be shared across all the protocol families and logical interfaces configured on a physical interface. This single policer is referenced in one or more firewall filters, and the filters, which are defined for a specific protocol family, are then applied to one or more logical interfaces configured on the physical interface. As a result, a single physical interface policer can apply to multiple routing instances because that policer includes all the logical interfaces and protocol families configured on the physical interface even if they belong to different instances. This feature is useful when you want to perform aggregate policing for different protocol families and different logical interfaces on the same physical interface. For example, a provider edge (PE) router has numerous logical interfaces, each corresponding to a different customer, configured on the same link to a customer edge (CE) device. A customer wants to apply rate limits aggregately on a single physical interface for certain types of traffic. A single aggregate policer for the physical interface would include all the logical interfaces configured and apply to all the routing instances to which those interfaces belong.

Physical interface policing is defined within a firewall filter for each protocol family. The supported protocol families include IPv4, IPv6, VPLS, MPLS, and circuit cross-connect (ccc). The physical interface policer is also applied an action to each firewall filter term that references the policer. That firewall filter is then applied on a logical interface as an output or input filter.

The following limitations apply:

- You cannot apply a firewall filter that references a physical interface policer to logical interfaces that do not belong to the physical interface for which the policer has been defined.
- You cannot define a firewall filter as both a physical interface filter and as a logical interface filter using the **interface-specific** statement.
- You cannot define a firewall filter configured with **family any** as a physical interface filter. A physical interface firewall filter must be defined for a specific protocol family.
- A firewall filter that is defined as physical interface filter must reference a physical interface policer. The filter cannot reference policer configured with the **interface-specific** statement.

Related Topics ■ [Configuring Physical Interface Policers on page 273](#)

Configuring Physical Interface Policers

A physical interface policer defines rate-limiting parameters for all the logical interfaces and protocol families configured on a physical interface. These logical interfaces can belong to different routing instances. You reference the policer within one or more firewall filters. You must also apply the physical interface policer as an action for each term used to define a set of match conditions for traffic on which you want to

perform rate limiting. You apply the firewall filters as input or output filters to the logical interfaces configured on the physical interface referenced in the policer.

The following sections describe how to configure a physical interface policer, reference the policer within a firewall filter, apply the policer as an action for a firewall filter, and apply (to a logical interface) a firewall filter that references a physical interface filter.

- Configuring Physical Interface Policers on page 274
- Configuring Firewall Filters That Reference Physical Interface Policers on page 275
- Applying Firewall Filters That Reference Physical Interface Policers on page 276

Configuring Physical Interface Policers

To configure a policer for a physical interface:

1. Include the `physical-interface-policer` statement at the `[edit firewall policer policer-name]` hierarchy level.
2. Include the `if-exceeding` statement at the `[edit firewall policer policer-name]` hierarchy level to define rate-limiting parameters for the policer.

For the `if-exceeding` statement, you must configure the following parameters:

- `bandwidth-limit bps`—Traffic rate, in bits per second (bps)
 - `burst-size-limit bytes`—Maximum burst size, in bytes
3. Include the `then policer-action` statement at the `[edit firewall policer policer-name]` hierarchy level to apply an action to the policer.

For `policer-action`, you can apply the following:

- `discard`—Discard a packet that exceeds the rate limits
- `loss-priority level`—Set the loss priority level to `low`, `medium-low`, `medium-high`, `high`.
- `forwarding-class class-name`—Specify the forwarding class for any `class-name` already configured.

In the following example, a physical interface policer, `shared-police1`, is configured to rate-limit traffic at 10000000000 bps and to permit a maximum burst of traffic of 500000 bytes. The `discard` action results in the discarding of packets that exceed the configured rate limits.

```
[edit]
firewall {
  policer shared-police1 {
    physical-interface-policer;
    if-exceeding {
      bandwidth-limit 100m;
      burst-size-limit 500k;
    }
    then {
```

```

        discard;
    }
}

```

Configuring Firewall Filters That Reference Physical Interface Policers

To use a physical interface policer, you must reference it in a firewall filter. For each filter, you also configure one or more terms for which you configure match conditions to define the types of traffic on which you limit traffic. To apply the policer to traffic that meets the match conditions in a term, you configure the physical interface policer as an action for the term.

To configure a firewall filter that references a physical interface policer:

1. Include the `physical-interface-filter` statement at the [edit firewall family *family-name* filter *filter-name*] hierarchy level.



NOTE: You cannot specify `family any`. You must configure a specific protocol family for a firewall filter that references a physical interface policer.

2. Include the `term term-name` statement at the [edit firewall filter family *family-name* filter *filter-name*] hierarchy level to define a term.
3. Include the `from match-conditions` statement at the [edit firewall family *family-name* filter *filter-name* term *term-name*] hierarchy level to define the characteristics that packets must have to have rate limiting performed as defined in the physical interface policer.

For more information about configuring specific match conditions, see Overview of Match Conditions in Firewall Filter Terms.

4. Include the `then policer policer-name` statement at the [edit firewall family *family-name* filter *filter-name* term *term-name*] hierarchy level to apply the specified physical interface policer as an action for the specified term. The rate-limiting parameters defined in the physical interface policer are performed on any traffic that matches the conditions defined in the term.

In the following example, a firewall filter is configured that references a physical interface filter. The filter is configured with `family inet` as the protocol family. A term `tcp-police-1` is defined to match any IPv4 traffic that is received through TCP with the IP precedence fields `critical-ecp`, `immediate`, or `priority`. IPv4 traffic that matches these characteristics has rate limiting performed, as defined in the `shared-police1` policer, which is applied as an action to the term `tcp-police-1`. A second term, `tcp-police-2`, is defined to match IPv4 traffic received through TCP with the IP precedence fields `internet-control` or `routine`. IPv4 traffic that matches these characteristics has rate limiting performed, as defined in the `shared-police1` policer, which is applied as an action to the term `tcp-police-2`.

```

[edit firewall]
family inet {
    filter inet-filter {

```

```

physical-interface-filter;
term tcp-police-1 {
    from {
        precedence [ critical-ecp immediate priority ];
        protocol tcp;
    }
    then policer shared-police1;
}
term tcp-police-2 {
    from {
        precedence [ internet-control routine ];
        protocol tcp;
    }
    then policer shared-police1
}
}
}

```

Applying Firewall Filters That Reference Physical Interface Policers

After you configure a firewall filter that references a physical interface policer, you apply it as an input or an output filter to a logical interface.

To apply a firewall filter that references a physical interface policer as an input filter:

- Include the `input filter-name` statement at the [edit interfaces *interface-name* unit *logical-unit-number* family *family-name* filter] hierarchy level.

To apply a firewall filter that references a physical interface policer as an output filter:

- Include the `output filter-name` statement at the [edit interfaces *interface-name* unit *logical-unit-number* family *family-name*] hierarchy level.

In the following example, firewall filter `inet-filter` is applied to family `inet` on interface `ge-1/2/0.0`. The filter is applied to incoming IPv4 traffic on the interface.

```

[edit]
interfaces {
    ge-1/2/0 {
        unit 0 {
            family inet {
                filter {
                    input inet-filter;
                }
                address 10.100.16.2/24
            }
        }
    }
}

```

Configuring Bandwidth Policers

The JUNOS Software supports policers that rate-limit traffic based on a percentage of physical port speed on an interface.

A bandwidth policer provides similar rate limiting at the logical interface level. For a bandwidth policer, the rate-limiting policer is based on a percentage of the configured logical interface bandwidth, defined as the shaping rate on that logical interface configured with class-of-service statements.

You can configure a policer to limit the bandwidth and apply that policer to multiple logical interfaces.

To configure a bandwidth policer, include the `logical-bandwidth-policer` statement at the `[edit firewall policer policer-name]` hierarchy level:

```
logical-bandwidth-policer;
```

You can configure rate limiting on the logical interface policer. For information about configuring rate limiting, see “Configuring Rate Limiting” on page 260. You can configure a policer action for the logical interface policer. For information about configuring policy actions, see “Configuring Policer Actions” on page 261.

After configuring the bandwidth policer, you can apply the policer to an interface. To apply a bandwidth policer to a logical interface, include the `policer policer-name` statement at the `[edit interfaces interface-name unit 0 family family-name]` hierarchy level:

```
policer (arp | input | output) policer-name;
```

For more information about applying policers, see the *JUNOS Class of Service Configuration Guide*.

Example: Configuring a Bandwidth Policer

Configure a bandwidth policer to rate-limit traffic for a logical interface:

```
[edit firewall policer new-police1]
if-exceeding {
  bandwidth-percent 10;
  burst-size-limit 125k;
}
logical-bandwidth-policer;
then {
  discard;
}
```

Apply the bandwidth policer to rate-limit IPv4 and IPv6 traffic on interface `fe-0/1/1`:

```
[edit interfaces fe-0/1/1 unit 0 family inet]
policer input new-police1;
[edit interfaces fe-0/1/1 unit 0 family inet6]
```

```
policer output new-police1;
```

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 333.

Configuring Tricolor Marking

For T Series routers and M320 routers with Enhanced II Flexible PIC Concentrators (FPCs), you can configure single-rate or two-rate tricolor marking (TCM).

TCM extends the functionality of class-of-service (CoS) traffic policing by providing three levels of drop priority instead of two. This allows you to provision more enhanced service-level agreements (SLAs) across the Differentiated Services (DiffServ) domain by defining tricolor marking policers, and three levels of packet loss priority (PLP) for classifiers, rewrite rules, random early detection (RED) drop profiles, and firewall filters.

The color of a packet, as used or set by a tricolor marking policer, corresponds to the packet’s drop precedence (loss priority or PLP). Packets with high PLP are marked red, packets with medium PLP are marked yellow, and packets with low PLP are marked green.

The following sections describe tricolor marking policers:

- Configuring Tricolor Marking Policers on page 278
- Configuring Interface Policers Using Tricolor Marking Policing on page 280

Configuring Tricolor Marking Policers

A tricolor marking policer polices traffic on the basis of metering, including the committed information rate (CIR), the peak information rate (PIR), and their associated burst sizes.

To configure a tricolor marking policer, include the **three-color-policer** statement at the **[edit firewall]** hierarchy level:

```
[edit firewall]
three-color-policer (Configuring) name {
  single-rate {
    (color-aware | color-blind);
    committed-information-rate bps;
    committed-burst-size bytes;
    excess-burst-size bytes;
  }
  two-rate {
    (color-aware | color-blind);
    committed-information-rate bps;
    committed-burst-size bytes;
    peak-information-rate bps;
    peak-burst-size bytes;
  }
}
```

When you configure this type of policer, you can set up to three loss priorities: low, medium-high, and high.



NOTE: To configure a policer that marks packets so that they have medium-low loss priority, you must configure a policer at the [edit firewall policer *policer-name*] hierarchy level.

For example:

```
[edit firewall]
policer 4PLP {
  if-exceeding {
    bandwidth-limit 40k;
    burst-size-limit 4k;
  }
  then loss-priority medium-low;
}
```

Apply this policer at one or both of the following hierarchy levels:

- [edit firewall family *family* filter *filter-name* term *rule-name* then policer *policer-name*]
- [edit interfaces *interface-name* unit *logical-unit-number* family *family* filter]

Specify the **single-rate** statement to configure marking based on CIR. If a packet exceeds the CIR in a single-rate policer, it is evaluated by the CBS. Specify the **committed-burst-size** option value to configure the maximum number of bytes allowed for incoming packets to burst above the CIR, but still be marked green. Specify the **excess-burst-size** option value to configure the maximum number of bytes allowed for incoming packets to burst above the CIR, but be marked red.

Specify the **two-rate** statement to configure marking based on CIR and PIR. If a packet exceeds the CIR in a two-rate policer, it is evaluated by the PIR. Specify the **committed-information-rate** option value to configure the guaranteed bandwidth under normal line conditions, and the rate up to which packets are marked green. Specify

the **committed-burst-size** option value to configure the maximum number of bytes allowed for incoming packets to burst above the CIR, but still be marked green.

Specify the **peak-information-rate** option value to configure the maximum achievable rate. Packets that exceed the CIR, but are below the PIR, are marked yellow. Packets that exceed the PIR are marked red. Specify the **peak-burst-size** option value to configure the maximum number of bytes allowed for incoming packets to burst above the PIR, but still be marked yellow.

For both the **single-rate** statement and the **two-rate** statement, specify the **color-aware** option value to configure metering by preclassification. Metering can increase a PLP, but cannot decrease it. Specify the **color-blind** option value to ignore any preclassification.

For more information about tricolor marking, see the *JUNOS Class of Service Configuration Guide*.

Example: Configuring a Tricolor Marking Policer

Configure a tricolor policer:

```
[edit firewall]
three-color-policer trtcm1 {
  two-rate {
    color-blind;
    committed-information-rate 1048576;
    committed-burst-size 65536;
    peak-information-rate 10485760;
    peak-burst-size 131072;
  }
}
```

Apply the tricolor policer to a firewall filter.

```
[edit firewall]
filter fil {
  term default {
    then {
      three-color-policer {
        two-rate trtcm1;
      }
    }
  }
}
```

Configuring Interface Policers Using Tricolor Marking Policing

You can configure a policer to limit traffic on an interface in the ingress or egress direction. Instead of policing each address family individually on an interface, you can aggregate policing with one policer. This single aggregated policer is known as the logical-interface policer. You can configure tricolor marking policing to limit the bandwidth through a logical interface.

To configure a policer on a logical interface using tricolor marking policing, include the `action` statement and the `logical-interface-policer` statement at the `[edit firewall three-color-policer name]` hierarchy level:

```
[edit firewall]
three-color-policer policer-name {
  action {
    loss-priority high then discard;
  }
  logical-interface-policer;
  single-rate {
    (color-aware | color-blind);
    committed-information-rate bps;
    committed-burst-size bytes;
    excess-burst-size bytes;
  }
  two-rate {
    (color-aware | color-blind);
    committed-information-rate bps;
    committed-burst-size bytes;
    peak-information-rate bps;
    peak-burst-size bytes;
  }
}
```

For detailed information about bandwidth policers on a logical interface, see “Configuring Aggregate Policers” on page 271.

You can configure separate policing on the ingress and egress direction on the logical interface.

Example: Rate-Limiting Bandwidth Using Tricolor Marking Policing

Configure tricolor marking policing on a logical interface to rate-limit the bandwidth on the logical interface.

```
[edit firewall]
three-color-policer trtcm-1 {
  action {
    loss-priority high then discard;
  }
  logical-interface-policer;
  two-rate {
    color-blind;
    committed-information-rate 1500000;
    committed-burst-size 150k;
    peak-information-rate 3m;
    peak-burst-size 300k;
  }
}
```

Examples: Configuring Policing

The following example shows a complete filter configuration containing a policer. It limits all FTP traffic from a given source to certain rate limits. Traffic exceeding the limits is discarded, and the remaining traffic is accepted and counted.

```
[edit]
firewall {
  policer policer-1 {
    if-exceeding {
      bandwidth-limit 400k;
      burst-size-limit 100k;
    }
    then {
      discard;
    }
  }
  term tcp-ftp {
    from {
      source-address 10.2.3/24;
      protocol tcp;
      destination-port ftp;
    }
    then {
      policer policer-1;
      accept;
      count count-ftp;
    }
  }
}
```

The following example shows a complete filter configuration containing two policers, and includes the **next term** action. Policer **policer-1** limits all traffic from a given source to certain rate limits, then sets the forwarding class. Policer **policer-2** limits all traffic to a second set of rate limits. Traffic exceeding the limits is discarded; the remaining traffic is accepted.

```
[edit]
firewall {
  policer policer-1 {
    if-exceeding {
      bandwidth-limit 10m;
      burst-size-limit 100k;
    }
    then {
      forwarding-class 0;
    }
  }
  policer policer-2 {
    if-exceeding {
      bandwidth-limit 100m;
      burst-size-limit 100k;
    }
    then {
```

```

        discard;
    }
}
filter f {
    term term-1 {
        then {
            policer policer-1;
            next term;
        }
    }
    term term-2 {
        then {
            policer policer-2;
            accept;
        }
    }
}
}

```

The following example limits all FTP traffic from a given source to certain rate limits, but defines the policer outside the filter, thereby creating a template that can be referenced by more than one filter or more than one term within a filter. Traffic exceeding the limits is discarded, and the remaining traffic is accepted and counted.

```

[edit]
firewall {
    policer policer-1 {
        if-exceeding {
            bandwidth-limit 400k;
            burst-size-limit 100k;
        }
        then {
            discard;
        }
    }
    filter limit-ftp {
        term tcp-ftp {
            from {
                source-address 10.2.3/24;
                protocol tcp;
                destination-port ftp;
            }
            then {
                policer policer-1;
                accept;
                count count-ftp;
            }
        }
    }
}
}

```

The following example shows a filter intended to thwart denial-of-service (DoS) SYN attacks:

```

[edit]
firewall {

```

```

    policer syn-recvd {
        if-exceeding {
            bandwidth-limit 40k;
            burst-size-limit 15000;
        }
        then discard;
    }
    term allow-syn {
        from {
            source-address {
                192.168.12.50/32; # trusted addresses
            }
        }
        then {
            log;
            accept;
        }
    }
    term limit-syn {
        from {
            protocol tcp;
            tcp-initial;
        }
        then {
            count limit-syn;
            policer syn-recvd;
            accept;
        }
    }
    term default {
        then accept;
    }
}
[edit] # apply filter to lo0 to control traffic to the Routing Engine
interfaces {
    lo0 {
        unit 0 {
            family inet {
                filter {
                    input syn-attack;
                }
            }
            address 172.16.4.53/32;
        }
    }
}

```

The following example uses one filter to do the following:

- Stop all User Datagram Protocol (UDP) and Internet Control Message Protocol (ICMP) traffic destined to these addresses (in **term a**).
- Send ICMP through the policer (in **term b**).
- Accept ICMP traffic within contract and all other traffic (in **term c**).



NOTE: It is important to keep the terms in order; once a packet has a match within the firewall filter, it is not examined in subsequent terms. For example, if you configured the filter to send ICMP traffic through the policer before discarding ICMP and UDP traffic to the addresses (in **term a**), you would not get the desired result.

```
[edit firewall]
policer policer-1 {
  if-exceeding {
    bandwidth-limit 200k;
    burst-size-limit 3k;
  }
  then {
    loss-priority high;
    forwarding-class 1;
  }
}
term a {
  from {
    destination-address {
      10.126.50.2/23;
      10.130.12.1/23;
      10.82.16.0/24 except;
      10.82.0.3/18;
    }
    protocol [icmp udp];
  }
  then {
    count packets-dropped;
    discard;
  }
}
term b {
  from {
    protocol icmp;
  }
  then policer policer-1;
}
term c {
  then accept;
}
```


Chapter 12

Summary of Firewall Filter and Policer Configuration Statements

The following descriptions explain each of the firewall filter and policer configuration statements. The statements are organized alphabetically.

accounting-profile

Syntax	accounting-profile <i>name</i> ;
Hierarchy Level	[edit firewall family <i>family-name</i> filter <i>filter-name</i>]
Release Information	Statement introduced before JUNOS Release 7.4.
Description	Enable collection of accounting data for the specified filter.
Options	<i>name</i> —Name assigned to the accounting profile.
Usage Guidelines	See “Configuring Accounting for Firewall Filters” on page 247.
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.

action

Syntax action {
 loss-priority high then discard;
 }

Hierarchy Level [edit firewall three-color-policer *name*],
 [edit logical-systems *logical-system-name* firewall three-color-policer *name*]

Release Information Statement introduced in JUNOS Release 8.2.
 Logical systems support introduced in JUNOS Release 9.3.

Description Discard traffic on a logical interface using tricolor marking policing.



NOTE: This statement is supported only on IQ2 interfaces.

Options The statements are explained separately in this chapter.

Usage Guidelines See “Configuring Interface Policers Using Tricolor Marking Policing” on page 280.

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

family

Syntax

```
family family-name {
    filter filter-name {
        accounting-profile name;
        interface-specific;
        physical-interface-filter;
    }
    prefix-action name {
        count;
        destination-prefix-length prefix-length;
        policer policer-name;
        source-prefix-length prefix-length;
        subnet-prefix-length prefix-length;
    }
    simple-filter filter-name {
        term term-name {
            from {
                match-conditions;
            }
            then {
                action;
                action-modifiers;
            }
        }
    }
}
```

Hierarchy Level [edit firewall],
[edit logical-systems *logical-system-name* firewall]

Release Information Statement introduced before JUNOS Release 7.4.
Logical systems support introduced in JUNOS Release 9.3.
simple-filter statement introduced in JUNOS Release 7.6.
any family type introduced in JUNOS Release 8.0.
bridge family type introduced in JUNOS Release 8.4 (MX Series routers only).

Description Configure a firewall filter for IP version 4 (IPv4) or IP version 6 (IPv6) traffic. On the MX Series routers only, configure a firewall filter for Layer 2 traffic in a bridging environment.

Options *family-name*—Version or type of addressing protocol:

- any—Protocol-independent match conditions.
- bridge—(MX Series routers only) Layer 2 packets that are part of bridging domain.
- ccc—Layer 2 switching cross-connects.
- inet—IPv4 addressing protocol.
- inet6—IPv6 addressing protocol.
- mpls—MPLS.

- **vpls**—Virtual private LAN service (VPLS).

The remaining statements are explained separately.

Usage Guidelines See Configuring Standard Firewall Filters and Example: Configuring CoS for a PBB Network on MX Series Routers.

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

filter

Syntax

```
filter filter-name {
    accounting-profile name;
    interface-specific;
    physical-interface-filter;
    term term-name {
        filter filter-name;
        from {
            match-conditions;
        }
        then {
            action;
            action-modifiers;
        }
    }
}
```

Hierarchy Level [edit firewall family *family-name*],
[edit logical-systems *logical-system-name* firewall family *family-name*]

Release Information Statement introduced before JUNOS Release 7.4.
Logical systems support introduced in JUNOS Release 9.3.
physical-interface-filter statement introduced in JUNOS Release 9.6.

Description Configure firewall filters.

Options *filter-name*—Name that identifies the filter. The name can contain letters, numbers, and hyphens (-) and can be up to 64 characters long. To include spaces in the name, enclose it in quotation marks (" ").

The remaining statements are explained separately.

Usage Guidelines See Configuring Standard Firewall Filters and Example: Configuring CoS for a PBB Network on MX Series Routers.

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

filter-specific

Syntax	filter-specific;
Hierarchy Level	[edit firewall policer <i>policer-name</i>], [edit logical-systems <i>logical-system-name</i> firewall policer <i>policer-name</i>]
Release Information	Statement introduced before JUNOS Release 7.4. Logical systems support introduced in JUNOS Release 9.3.
Description	Configure a policer to act as a filter-specific policer. If this statement is not specified, then the policer defaults to a term-specific policer.
Usage Guidelines	See “Configuring Filter-Specific Policers” on page 263.
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.

firewall

Syntax	firewall { ... }
Hierarchy Level	[edit], [edit logical-systems <i>logical-system-name</i>]
Release Information	Statement introduced before JUNOS Release 7.4. Logical systems support introduced in JUNOS Release 9.3.
Description	Configure firewall filters. The statements are explained separately.
Usage Guidelines	See “Configuring Standard Firewall Filters” on page 179.
Required Privilege Level	firewall—To view this statement in the configuration. firewall-control—To add this statement to the configuration.

if-exceeding

Syntax if-exceeding {
 bandwidth-limit *bps*;
 bandwidth-percent *number*;
 burst-size-limit *bytes*;
 }

Hierarchy Level [edit firewall policer *policer-name*],
 [edit logical-systems *logical-system-name* firewall policer *policer-name*]

Release Information Statement introduced before JUNOS Release 7.4.
 Logical systems support introduced in JUNOS Release 9.3.

Description Configure policer rate limits.

Options bandwidth-limit *bps*—Traffic rate, in bits per second (bps). Any value below 61,040 bps results in an effective rate of 30,520 bps.
Range: 8000 through 40,000,000,000 bps



NOTE: You can configure a minimum of 8000 bps on the MX Series, M120, and M320 routers only.

Range: 32,000 through 40,000,000,000 bps



NOTE: The minimum value that you can configure on any platform except for the MX Series, M120, and M320 routers is 32,000 bps.

Default: None

bandwidth-percent *number*—Port speed, in decimal percentage number.

Range: 1 through 100

Default: None

burst-size-limit *bytes*—Maximum burst size. The minimum recommended value is the maximum transmission unit (MTU) of the IP packets being policed.

Range: 1500 through 100,000,000,000 bytes

Default: None

Usage Guidelines See “Configuring Rate Limiting” on page 260.

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

interface-set

Syntax	<code>interface-set <i>interface-set-name</i> { <i>interface-name</i>; }</code>
Hierarchy Level	[edit firewall], [edit logical-systems <i>logical-system-name</i> firewall]
Release Information	Statement introduced before JUNOS Release 7.4. Logical systems support introduced in JUNOS Release 9.3.
Description	Configure an interface set.
Options	<i>interface-name</i> —Names of each interface to include in the interface set. You must specify more than one name.
Usage Guidelines	See “Configuring Interface Sets” on page 269.
Required Privilege Level	firewall—To view this statement in the configuration. firewall-control—To add this statement to the configuration.

interface-specific

Syntax	<code>interface-specific;</code>
Hierarchy Level	[edit firewall family <i>family-name</i> filter <i>filter-name</i>], [edit logical-systems <i>logical-system-name</i> firewall family <i>family-name</i> filter <i>filter-name</i>]
Release Information	Statement introduced before JUNOS Release 7.4. Logical systems support introduced in JUNOS Release 9.3.
Description	Configure interface-specific names for firewall counters.
Usage Guidelines	See “Configuring Interface-Specific Counters” on page 216.
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.

load-balance-group

Syntax	<code>load-balance-group <i>group-name</i> { next-hop-group [<i>group-names</i>]; }</code>
Hierarchy Level	[edit firewall]
Release Information	Statement introduced before JUNOS Release 7.4.
Description	Configure a load-balance group.
Options	<i>group-name</i> —Name of load-balance group. <i>group-names</i> —Name of next-hop groups to include in the load-balance group set.
Usage Guidelines	See “Configuring Load-Balance Groups” on page 278.
Required Privilege Level	firewall—To view this statement in the configuration. firewall-control—To add this statement to the configuration.

logical-bandwidth-policer

Syntax	<code>logical-bandwidth-policer;</code>
Hierarchy Level	[edit firewall policer <i>policer-name</i>], [edit logical-systems <i>logical-system-name</i> firewall policer <i>policer-name</i>]
Release Information	Statement introduced in JUNOS Release 8.2. Logical systems support introduced in JUNOS Release 9.3.
Description	Extend the policer rate limits to logical interfaces. The policer rate limit is based on the shaping rate defined on the logical interface.
Usage Guidelines	See “Configuring Bandwidth Policers” on page 277.
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.

logical-interface-policer

Syntax	logical-interface-policer;
Hierarchy Level	[edit firewall policer <i>policer-name</i>], [edit firewall three-color-policer <i>name</i>], [edit logical-systems <i>logical-system-name</i> firewall policer <i>policer-name</i>], [edit logical-systems <i>logical-system-name</i> firewall three-color-policer <i>name</i>]
Release Information	Statement introduced before JUNOS Release 7.4. Support at the [edit firewall three-color-policer <i>name</i>] hierarchy level introduced in JUNOS Release 8.2. Logical systems support introduced in JUNOS Release 9.3.
Description	Configure an aggregate policer.
Usage Guidelines	See “Configuring Aggregate Policers” on page 271 and “Configuring Tricolor Marking” on page 278.
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.

physical-interface-filter

Syntax	physical-interface-filter;
Hierarchy Level	[edit firewall family <i>family-name</i> filter <i>filter-name</i>], [edit logical-systems <i>logical-system-name</i> firewall family <i>family-name</i> filter <i>filter-name</i>], [edit routing-instances <i>routing-instance-name</i> firewall family <i>family-name</i> filter <i>filter-name</i>], [edit logical-systems <i>logical-system-name</i> routing-instances <i>routing-instance-name</i> firewall family <i>family-name</i> filter <i>filter-name</i>]
Release Information	Statement introduced in JUNOS Release 9.6.
Description	Configure a physical-interface filter. Use this statement to reference a physical-interface policer for the specified protocol family.
Usage Guidelines	See “Configuring Physical Interface Policers” on page 273
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.
Related Topics	physical-interface-policer, policer

physical-interface-policer

Syntax	physical-interface-policer;
Hierarchy Level	[edit firewall policer <i>policer-name</i>], [edit logical-system <i>logical-system-name</i> firewall policer <i>policer-name</i>], [edit routing-instances <i>routing-instance-name</i> firewall policer <i>policer-name</i>], [edit logical-systems <i>logical-system-name</i> routing-instances <i>routing-instance-name</i> firewall policer <i>policer-name</i>]
Release Information	Statement introduced in JUNOS Release 9.6.
Description	Configure an aggregate policer for a physical interface. A physical-interface policer applies to all the logical interfaces and protocol families configured on a physical interface. As result, a single physical-interface policer can be applied to multiple routing instances because this policer includes all the logical interfaces configured on the physical interface even if they belong to different routing instances.
Usage Guidelines	See “Configuring Physical Interface Policers” on page 273
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.
Related Topics	physical-interface-filter

policer

Syntax	<pre> policer <i>policer-name</i> { filter-specific; if-exceeding { bandwidth-limit <i>bps</i>; bandwidth-percent <i>number</i>; burst-size-limit <i>bytes</i>; } logical-interface-policer; physical-interface-policer; then { <i>policer-action</i>; } } </pre>
Hierarchy Level	[edit firewall], [edit logical-systems <i>logical-system-name</i> firewall]
Release Information	Statement introduced before JUNOS Release 7.4. Logical systems support introduced in JUNOS Release 9.3. <i>physical-interface-policer</i> statement introduced in JUNOS Release 9.6.
Description	Configure policer rate limits and actions. When included at the [edit firewall] hierarchy level, it creates a template, and you do not have to configure a policer individually for every firewall filter or interface. To activate a policer, you must include the policer action modifier in the then statement in a firewall filter term or on an interface.
Options	<p><i>policer-action</i>—One or more actions to take:</p> <ul style="list-style-type: none"> ■ discard—Discard traffic that exceeds the rate limits. ■ forwarding-class <i>class-name</i>—Specify the particular forwarding class. ■ loss-priority—Set the packet loss priority (PLP) to low, medium-low, medium-high, or high. <p><i>policer-name</i>—Name that identifies the policer. 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 (" ").</p> <p>then—Actions to take on matching packets.</p> <p>The remaining statements are explained separately.</p>
Usage Guidelines	See “Configuring Policers” on page 259.
Required Privilege Level	firewall —To view this statement in the configuration. firewall-control —To add this statement to the configuration.
Related Topics	<ul style="list-style-type: none"> ■ Example: Configuring CoS for a PBB Network on MX Series Routers

- physical-interface-policer

prefix-action

Syntax prefix-action *name* {
 count;
 destination-prefix-length *prefix-length*;
 policer *policer-name*;
 source-prefix-length *prefix-length*;
 subnet-prefix-length *prefix-length*;
 }

Hierarchy Level [edit firewall family inet],
 [edit logical-systems *logical-system-name* firewall family inet]

Release Information Statement introduced before JUNOS Release 7.4.
 Logical systems support introduced in JUNOS Release 9.3.

Description Configure prefix-specific action.

Options count—Enable counter.

destination-prefix-length *prefix-length*—Destination prefix length.

Range: 0 through 32

policer *policer-name*—Policer name.

source-prefix-length *prefix-length*—Source prefix length.

Range: 0 through 32

subnet-prefix-length *prefix-length*—Subnet prefix length.

Range: 0 through 32

Usage Guidelines See “Configuring Policer Actions for Specific Address Prefixes” on page 264.

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

service-filter

Syntax	<pre> service-filter <i>filter-name</i> { term <i>term-name</i> { from { <i>match-conditions</i>; } then { <i>action</i>; <i>action-modifiers</i>; } } } </pre>
Hierarchy Level	[edit firewall family inet], [edit logical-systems <i>logical-system-name</i> firewall family inet]
Release Information	Statement introduced before JUNOS Release 7.4. Logical systems support introduced in JUNOS Release 9.3.
Description	Configure service filters.
Options	<p><i>filter-name</i>—Name that identifies the service filter. 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 (" ").</p> <p>The remaining statements are explained separately.</p>
Usage Guidelines	See “Configuring Service Filters” on page 231.
Required Privilege Level	firewall—To view this statement in the configuration. firewall-control—To add this statement to the configuration

simple-filter

Syntax	<pre> simple-filter <i>filter-name</i> { term <i>term-name</i> { from { <i>match-conditions</i>; } then { <i>action</i>; <i>action-modifiers</i>; } } } </pre>
Hierarchy Level	[edit firewall family inet], [edit logical-systems <i>logical-system-name</i> firewall family inet]
Release Information	Statement introduced in JUNOS Release 7.6. Logical systems support introduced in JUNOS Release 9.3.
Description	Configure simple filters.
Options	<p><i>filter-name</i>—Name that identifies the simple filter. 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 (“ ”).</p> <p>The remaining statements are explained separately.</p>
Usage Guidelines	See “Configuring Simple Filters” on page 232.
Required Privilege Level	firewall—To view this statement in the configuration. firewall-control—To add this statement to the configuration

term

Syntax	<pre> term term-name { filter filter-name; from { match-conditions; } then { action; action-modifiers; } } </pre>
Hierarchy Level	<p>[edit firewall family <i>family-name</i> filter <i>filter-name</i>],</p> <p>[edit firewall family <i>family-name</i> service-filter <i>filter-name</i>],</p> <p>[edit firewall family <i>family-name</i> simple-filter <i>filter-name</i>],</p> <p>[edit logical-systems <i>logical-system-name</i> firewall family <i>family-name</i> filter <i>filter-name</i>],</p> <p>[edit logical-systems <i>logical-system-name</i> firewall family <i>family-name</i> service-filter <i>filter-name</i>],</p> <p>[edit logical-systems <i>logical-system-name</i> firewall family <i>family-name</i> simple-filter <i>filter-name</i>]</p>
Release Information	<p>Statement introduced before JUNOS Release 7.4.</p> <p>filter option introduced in JUNOS Release 7.6.</p> <p>Logical systems support introduced in JUNOS Release 9.3.</p>
Description	Define a firewall filter term.
Options	<p>actions—(Optional) An action to take if conditions match. If you do not specify an action, the packets that match the conditions in the from statement are accepted. The actions are described in Table 32 on page 209.</p> <p>action-modifiers—(Optional) One or more actions to perform on a packet. The action modifiers are described in Table 32 on page 209.</p> <p>filter-name—(Optional) A filter within a filter. This term references another filter.</p> <p>from—(Optional) Match packet fields to values. If not included, all packets are considered to match and the actions and action modifiers in the then statement are taken.</p> <p>match-conditions—One or more conditions to use to make a match. The conditions are described in Overview of Match Conditions in Firewall Filter Terms, Overview of Match Conditions in Firewall Filter Terms, and Overview of Match Conditions in Firewall Filter Terms.</p> <p>term-name—Name that identifies the term. The name can contain letters, numbers, and hyphens (-), and can be up to 64 characters long. To include spaces in the name, enclose it in quotation marks (" ").</p>

then—(Optional) Actions to take on matching packets. If not included and a packet matches all the conditions in the **from** statement, the packet is accepted.

Usage Guidelines See Configuring Standard Firewall Filters and Example: Configuring CoS for a PBB Network on MX Series Routers.

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

three-color-policer

See the following sections:

- three-color-policer (Applying) on page 303
- three-color-policer (Configuring) on page 304

three-color-policer (Applying)

Syntax three-color-policer {
 (single-rate | two-rate) *policer-name*;
 }

Hierarchy Level [edit firewall family *family-name* filter *filter-name* term *term-name* then],
 [edit logical-systems *logical-system-name* firewall family *family-name* filter *filter-name*
 term *term-name* then]

Release Information Statement introduced before JUNOS Release 7.4.
 single-rate statement added in JUNOS Release 8.2.
 Logical systems support introduced in JUNOS Release 9.3.

Description For T Series routers and M320 routers with Enhanced II Flexible PIC Concentrators (FPCs) and the T640 Core Router with Enhanced Scaling FPC4, apply a tricolor marking policer.

Options single-rate—Named tricolor policer is a single-rate policer.

two-rate—Named tricolor policer is a two-rate policer.

policer-name—Name of a tricolor policer.

Usage Guidelines See “Configuring Actions in Firewall Filter Terms” on page 208.

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

three-color-policer (Configuring)

Syntax	<pre> three-color-policer <i>policer-name</i> { action { loss-priority high then discard; } logical-interface-policer; single-rate { (color-aware color-blind); committed-information-rate <i>bps</i>; committed-burst-size <i>bytes</i>; excess-burst-size <i>bytes</i>; } two-rate { (color-aware color-blind); committed-information-rate <i>bps</i>; committed-burst-size <i>bytes</i>; peak-information-rate <i>bps</i>; peak-burst-size <i>bytes</i>; } } </pre>
Hierarchy Level	[edit firewall], [edit logical-systems <i>logical-system-name</i> firewall]
Release Information	Statement introduced before JUNOS Release 7.4. action statement introduced in JUNOS Release 8.2. Logical systems support introduced in JUNOS Release 9.3.
Description	Configure a tricolor marking policer.
Options	<p>color-aware—Metering varies by preclassification. Metering can increase a packet's assigned PLP, but cannot decrease it.</p> <p>color-blind—Packet preclassification is ignored. All packets are evaluated by the CBS. If a packet exceeds the CBS, it is evaluated by the EBS.</p> <p>committed-burst-size <i>bytes</i>—Maximum bytes allowed for incoming packets to be marked green. Range: 1500 through 100,000,000,000 bytes</p> <p>committed-information-rate <i>bps</i>—Guaranteed bandwidth under normal line conditions, and the average rate up to which packets are marked green. Range: 32,000 through 40,000,000,000 bps</p> <p>excess-burst-size <i>bytes</i>—Maximum bytes allowed for incoming packets. Packets that exceed the EBS are marked red. Range: 1500 through 100,000,000,000 bytes</p> <p>peak-burst-size <i>bytes</i>—Maximum bytes allowed for incoming packets to burst above the PIR, but still be marked yellow. Range: 1500 through 100,000,000,000 bytes</p>

peak-information-rate *bps*—Maximum achievable rate. Packets that exceed the CIR but are below the PIR are marked yellow. Packets that exceed the PIR are marked red.

Range: 32,000 through 40,000,000,000 bps

single-rate—Marking is based on the CIR, CBS, and the EBS.

two-rate—Marking is based on the CIR and the PIR.

The remaining statements are explained separately.

Usage Guidelines See “Configuring Tricolor Marking” on page 278.

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

virtual-channel

Syntax `virtual-channel virtual-channel-name;`

Hierarchy Level [edit firewall family *family-name* filter *filter-name* term *term-name* then],
[edit logical-systems *logical-system-name* firewall family *family-name* filter *filter-name* term *term-name* then]

Release Information Statement introduced before JUNOS Release 7.4.
Logical systems support introduced in JUNOS Release 9.3.

Description For J Series Services Routers only, select the traffic to be transmitted by way of a particular virtual channel. *virtual-channel-name* must be one of the names that you define at the [edit class-of-service virtual-channels] hierarchy level.

Options *virtual-channel-name*—Name of the virtual channel.

Usage Guidelines See the *JUNOS Class of Service Configuration Guide*.

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

Part 4

Traffic Sampling, Forwarding and Monitoring

- Traffic Sampling, Forwarding, and Monitoring Overview on page 309
- Introduction to Traffic Sampling Configuration on page 311
- Traffic Forwarding and Monitoring Configuration on page 325
- Summary of Traffic Sampling, Forwarding, and Monitoring Configuration Statements on page 349

Chapter 13

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 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 14

Introduction to Traffic Sampling Configuration

This chapter describes the following tasks for configuring traffic sampling:

- Traffic Sampling Configuration on page 311
- Minimum Traffic Sampling Configuration on page 312
- Configuring Traffic Sampling on page 313
- Disabling Traffic Sampling on page 315
- Configuring the Output File for Traffic Sampling on page 315
- Tracing Traffic Sampling Operations on page 317
- Configuring Flow Aggregation (cflowd) on page 317
- Configuring Active Flow Monitoring Using Version 9 on page 320
- Traffic Sampling Examples on page 321
- Example: Sampling a Single SONET/SDH Interface on page 321
- Example: Sampling All Traffic from a Single IP Address on page 322
- Example: Sampling All FTP Traffic on page 323

Traffic Sampling Configuration

To configure traffic sampling, include the **sampling** statement at the [edit forwarding-options] hierarchy level:

```
[edit forwarding-options]
sampling {
  disable;
  input {
    family (inet | mpls) {
      max-packets-per-second number;
      rate number;
      run-length number;
    }
  }
  output {
    cflowd hostname {
      version9 {
        template template-name;
      }
    }
  }
}
```

```

    }
    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;
}
file {
    disable;
    filename filename;
    files number;
    size bytes;
    (stamp | no-stamp);
    (world-readable | no-world-readable);
}
flow-active-timeout seconds;
flow-inactive-timeout seconds;
interface interface-name {
    engine-id number;
    engine-type number;
    source-address address;
}
}
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 {
        family inet {
            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 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.

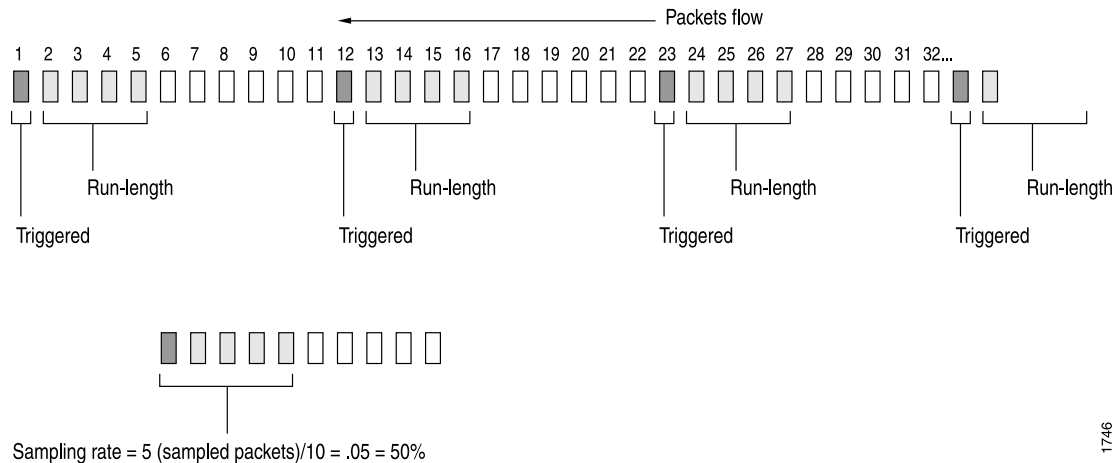
Specify the maximum length of the sampled packet by using the `maximum-packet-length bytes` statement. For *bytes*, specify a value from 0 through 9192.

Specify the sampling rate by setting the values for `rate` and `run-length` (see Figure 13 on page 314).

Figure 13: Configure Sampling Rate

Rate and Run-length

Case #1 Rate =10, run-length =4



1746

The `rate` statement specifies the ratio of packets to be sampled. For example, if you configure a rate of 10, x number of packets out of every 10 is sampled, where $x = \text{run-length} + 1$. By default, the rate is 0, which means that no traffic 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.

If you do not include the `input` statement, sampling is disabled.

To collect the sampled packets in a file, include the `file` statement at the `[edit forwarding-options sampling output]` hierarchy level. For more information about the output file formats, see “Configuring the Output File for Traffic Sampling” on page 315.

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 317 and “Configuring Active Flow Monitoring Using Version 9” on page 320.

The JUNOS Software 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. For more detailed information about configuring firewall filters, see "Firewall Filter Configuration" on page 177.

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 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 output family (inet | inet6 | mpls)]
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 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 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 320.

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
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 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 `cflowd` statement at the `[edit forwarding-options sampling output family family-name]` hierarchy level:

```
[edit forwarding-options sampling output family family-name]
cflowd 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 output cflowd 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 Software complies with Version 2.1b1 of *cflowd*. If you do not include the **caida-compliant** statement in the configuration, the JUNOS Software 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 output cflowd 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 315.



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 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 Services Interfaces Configuration Guide* and the *JUNOS Feature Guide*.

To enable the collection of traffic flows using the version 9 format, include the `version9` statement at the `[edit forwarding-options sampling output family family-name cflowd hostname]` hierarchy level:

```

[edit forwarding-options sampling output family family-name cflowd 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 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 Release 9.5 and later, you can sample sample packets both the `inet mpls` protocol families at the same time. For more information about how to configure traffic sampling, see “Configuring Traffic Sampling” on page 313.

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 output cflowd 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 317.

- 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 `[edit services monitoring]` hierarchy level. You also configure the traffic family `mpls` to sample.

```
[edit forwarding-options]
sampling {
  input {
    family mpls {
      rate 1;
      run-length;
    }
  }
  output {
    cflowd 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 321
- Example: Sampling All Traffic from a Single IP Address on page 322
- Example: Sampling All FTP Traffic on page 323

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 {
    family inet {
      rate 100;
      run-length 2;
    }
  }
  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 {
        family inet {
            rate 1;
        }
    }
    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 {
    family inet {
      rate 10;
    }
  }
  output {
    file {
      filename t3-ftp-traffic.txt;
      files 50;
      size 1m;
    }
  }
}

```

Chapter 15

Traffic Forwarding and Monitoring Configuration

This chapter describes the following tasks for configuring forwarding options and traffic monitoring:

- Configuring Traffic Forwarding and Monitoring on page 325
- Applying Filters to Forwarding Tables on page 329
- Configuring IPv6 Accounting on page 330
- Configuring Discard Accounting on page 330
- Configuring Flow Monitoring on page 332
- Configuring Next-Hop Groups on page 333
- Per-Flow and Per-Prefix Load Balancing Overview on page 333
- Configuring Per-Prefix Load Balancing on page 334
- Configuring Per-Flow Load Balancing Based on Hash Values on page 335
- Configuring Routers, Switches, and Interfaces as DHCP and BOOTP Relay Agents on page 335
- Configuring DNS and TFTP Packet Forwarding on page 337
- Preventing DHCP Spoofing on MX Series Ethernet Services Routers on page 340
- Configuring Port Mirroring on page 341
- Configuring Packet Capture on page 345

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;
}
}
}
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 [ addresses ];
        }
        maximum-hop-count number;
        minimum-wait-time seconds;
        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-next-hop;
    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;
            interface interface-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;
}
}

```


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 250.

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 MX Series Ethernet Services Routers 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 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 is disabled after a reboot of the routing platform. To view IPv6 statistics, issue the `show interface statistics operational` mode command. For more information, see the *JUNOS 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** 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 interface-name {
      engine-id number;
      engine-type number;
      source-address address;
    }
  }
}
```

To configure an accounting group, include the **accounting** statement and specify a *group-name*. To configure the output flow aggregation, include the **cflowd** statement. For more information about flow aggregation, see “Configuring Flow Aggregation (cflowd)” on page 317. 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 “Firewall Filter Configuration” on page 177.

You can use discard accounting for passive and active flow monitoring. For more detailed information about configuring passive and active flow monitoring, see the *JUNOS Feature Guide* and the *JUNOS 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 317. 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 "Firewall Filter Configuration" on page 177.

For more detailed information about configuring passive and active flow monitoring, see the *JUNOS Feature Guide* and the *JUNOS Class of Service Configuration Guide*.

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. For more information about configuring port mirroring, see "Configuring Port Mirroring" on page 341 and the *JUNOS Feature Guide*.

Per-Flow and Per-Prefix Load Balancing Overview

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

On all M Series Multiservice Edge Routers, MX Series Ethernet Services Routers, and T Series Core Routers, you have two additional options:

- You can specify what information the routing platform uses for per-flow load balancing based on port data (instead of on source and destination IP addresses only). For aggregated Ethernet and aggregated SONET/SDH interfaces, you can load-balance based on the MPLS label information. For more information, see “Overview of Per-Packet Load Balancing” on page 144.
- You can also configure per-prefix load balancing, which allows you to configure a hash value that enables the router to elect a next hop independently of the route chosen by other routers. For more information, see “Configuring Per-Prefix Load Balancing” on page 334.

In addition, on the M120, M320, and MX Series routers only, you have the following option:

- You can also configure per-flow load balancing, which allows you to configure the router to assign a unique, load-balance hash value for each Packet Forwarding Engine slot. For more information, see “Configuring Per-Flow Load Balancing Based on Hash Values” on page 335.

Configuring Per-Prefix Load Balancing

By default, the JUNOS Software 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-next-hop;
  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.

To generate a permuted index of next-hop entries for unicast and aggregate next hops, include the `indexed-next-hop` statement at the `[edit forwarding-options load-balance]` hierarchy level:

```
indexed-next-hop;
```

Configuring Per-Flow Load Balancing Based on Hash Values

By default, the JUNOS Software 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-next-hop;
  per-flow {
    hash-seed;
  }
}
```

To enable per-flow load balancing, you must include the `hash-seed` statement. The JUNOS Software 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-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 [ routing-instance-names ]>;
    }
  }
  maximum-hop-count number;
  minimum-wait-time seconds;
  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 or addresses that specify the DHCP or BOOTP server for the router, switch, or interface, include the **server** statement. You can include as many addresses as necessary in the same statement.

To set an IP time-to-live (TTL) value for DHCP response packets sent to a DHCP client, include the **client-response-ttl** statement.

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 System Basics Configuration Guide*.

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 address ;
      server logical-system name < [ routing-instance routing-instance-names ] >;
      server < [ routing-instance routing-instance-names ] >;
    }
  }
  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 338
- Example: Configuring DNS Packet Forwarding on page 340

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 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 339
- Configuring the Number and Size of Log Files on page 339
- Configuring Access to the Log File on page 339
- Configuring a Regular Expression for Lines to Be Logged on page 340

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 Ethernet Services 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. For an example of DHCP snooping on the MX Series router, see the *JUNOS MX Series Ethernet Services 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 333.

All M Series Multiservice Edge Routers, T Series Core Routers, and MX Series Ethernet Services 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.

Port mirroring for VPLS traffic is supported on M7i and M10i routers configured with Enhanced CFEBs (CFEB-Es), on M120 routers, on M320 routers configured with Enhanced III Flexible PIC Concentrators (FPCs), and MX Series routers.

In JUNOS 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 MX-series Layer 2 Configuration Guide*.

In JUNOS 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 FPCs. 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.
- You cannot configure firewall filters on the port-mirroring interface.
- 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.
- 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. For more information about tunnel interfaces, see the *JUNOS Network Interfaces Configuration Guide*.

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 313.

Configuring Multiple Port-Mirroring Instances

In JUNOS 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 MX-series 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 (inet | inet6 | vpls | ccc) {
    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 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 High Availability Configuration Guide*. For information about configuring FPC to FEB connectivity on an M120 router, see the *JUNOS System Basics Configuration Guide*.

Configuring MX Series Ethernet Services 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 **var/tmp/**, the file is not recreated upon the next packet capture traffic on the interface. You must first disable and then enable PCAP functionality again to recreate 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: 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. For information about configuring interfaces, see the *JUNOS Network Interfaces Configuration Guide*.

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.

For more information about configuring specific interface types, see the *JUNOS Network Interfaces Configuration Guide*.

Chapter 16

Summary of Traffic Sampling, Forwarding, and Monitoring Configuration Statements

The following sections explain each of the sampling and forwarding statements. The statements are organized alphabetically.

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 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 Topics ■ Configuring Discard Accounting on page 330

aggregation

Syntax	<pre>aggregation { autonomous-system; destination-prefix; protocol-port; source-destination-prefix { caida-compliant; } source-prefix; }</pre>
Hierarchy Level	[edit forwarding-options accounting output cflowd <i>hostname</i>], [edit forwarding-options sampling output cflowd <i>hostname</i>]
Release Information	Statement introduced before JUNOS 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	<p>autonomous-system—Aggregate by autonomous system (AS) number.</p> <p>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 Software records source and destination mask length values in compliance with the <i>cflowd Configuration Guide</i>, dated August 30, 1999.</p> <p>destination-prefix—Aggregate by destination prefix.</p> <p>protocol-port—Aggregate by protocol and port number.</p> <p>source-destination-prefix—Aggregate by source and destination prefix.</p> <p>source-prefix—Aggregate by source prefix.</p>
Required Privilege Level	<p>interface—To view this statement in the configuration.</p> <p>interface-control—To add this statement to the configuration.</p>
Related Topics	<ul style="list-style-type: none"> ■ Configuring Flow Aggregation (cflowd) on page 317

autonomous-system-type

Syntax	autonomous-system-type (origin peer);
Hierarchy Level	[edit forwarding-options accounting output cflowd <i>hostname</i>], [edit forwarding-options sampling output cflowd <i>hostname</i>]
Release Information	Statement introduced before JUNOS Release 7.4.
Description	Specify the type of AS numbers that cflowd exports.
Options	<p>origin—Export origin AS numbers of the packet source address in the Source Autonomous System cflowd field.</p> <p>peer—Export peer AS numbers through which the packet passed in the Source Autonomous System cflowd field.</p> <p>Default: origin</p>
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.
Related Topics	■ Configuring Flow Aggregation (cflowd) on page 317

bootp

Syntax

```
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 [ routing-instance-names
        ]>;
    }
  }
  maximum-hop-count number;
  minimum-wait-time seconds;
  server address {
    <logical-system logical-system-name> <routing-instance [ routing-instance-names ]>;
  }
}
```

Hierarchy Level [edit forwarding-options helpers]

Release Information Statement introduced before JUNOS Release 7.4.
Statement introduced in JUNOS Release 9.0 for EX 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 Topics

- Configuring Routers, Switches and Interfaces as DHCP and BOOTP Relay Agents on page 335
- Setting Up DHCP Option 82 with the Switch as a Relay Agent Between Clients and DHCP Server (CLI Procedure)

cflowd

See the following sections:

- [cflowd \(Discard Accounting\) on page 355](#)
- [cflowd \(Flow Monitoring\) on page 356](#)
- [cflowd \(Sampling\) on page 357](#)

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;
 source-address address;
 version format;
}`

Hierarchy Level [edit forwarding-options accounting *group-name* output]

Release Information Statement introduced before JUNOS 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 Topics ■ Configuring Flow Aggregation (cflowd) on page 317

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 Release 7.4.

Description Collect an aggregate of sampled flows and send the aggregate to a specified host system that runs the collection utility `cfcollect`.

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 Topics ■ Configuring Flow Monitoring on page 332

cflowd (Sampling)

Syntax `cflowd 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 output]

Release Information Statement introduced before JUNOS Release 7.4.
 version9 statement introduced in JUNOS 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 cflowd *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 Topics

- Configuring Active Flow Monitoring Using Version 9 on page 320
- Configuring Flow Aggregation (cflowd) on page 317

client-response-ttl

Syntax	client-response-ttl <i>number</i> ;
Hierarchy Level	[edit forwarding-options helpers bootp], [edit forwarding-options helpers bootp interface <i>interface-group</i>]
Release Information	Statement introduced in JUNOS Release 8.1.
Description	Set the IP time-to-live (TTL) value in DHCP response packets sent to a DHCP client.
Options	<i>number</i> —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 Topics	■ Configuring Routers, Switches and Interfaces as DHCP and BOOTP Relay Agents on page 335

description

Syntax	description <i>text-description</i> ;
Hierarchy Level	[edit forwarding-options helpers bootp], [edit forwarding-options helpers bootp interface <i>interface-group</i>], [edit forwarding-options helpers domain], [edit forwarding-options helpers domain interface <i>interface-name</i>], [edit forwarding-options helpers tftp], [edit forwarding-options helpers tftp interface <i>interface-name</i>]
Release Information	Statement introduced before JUNOS Release 7.4. Statement introduced in JUNOS Release 9.0 for EX 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 Topics	■ Configuring DNS and TFTP Packet Forwarding on page 337 ■ Configuring Routers, Switches and Interfaces as DHCP and BOOTP Relay Agents on page 335

dhcp-relay (DHCP Spoofing Prevention)

Syntax	<pre>dhcp-relay { group group-name { interface interface-name; } }</pre>
Hierarchy Level	[edit routing-instances <i>routing-instance-name</i> bridge-domains <i>bridge-domain-name</i> forwarding-options], [edit routing-instances <i>routing-instance-name</i> forwarding-options]
Release Information	Statement introduced in JUNOS Release 9.4 (MX Series routers only).
Description	<p>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.</p> <p>Configure DHCP snooping by including the appropriate interfaces in the DHCP relay configuration.</p> <p>The statements are explained separately.</p>
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.
Related Topics	<ul style="list-style-type: none"> ■ Preventing DHCP Spoofing on MX Series Ethernet Services Routers on page 340

disable

Syntax	disable;
Hierarchy Level	[edit forwarding-options packet-capture], [edit forwarding-options sampling], [edit forwarding-options sampling output file]
Release Information	Statement introduced before JUNOS Release 7.4. Supported added at the [edit forwarding-options packet-capture] hierarchy level on J Series Services Routers in JUNOS 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 Topics	<ul style="list-style-type: none"> ■ Configuring Packet Capture on page 345 ■ Disabling Traffic Sampling on page 315

domain

Syntax	<pre> domain { description <i>text-description</i>; interface <i>interface-name</i> { broadcast; description <i>text-description</i>; no-listen; server address <logical-system <i>logical-system-name</i>> <routing-instance <i>routing-instance-name</i>>; } server address <logical-system <i>logical-system-name</i>> <routing-instance <i>routing-instance-name</i>>; } </pre>
Hierarchy Level	[edit forwarding-options helpers]
Release Information	Statement introduced before JUNOS Release 7.4. Statement introduced in JUNOS Release 9.0 for EX Series switches.
Description	<p>Enable DNS request packet forwarding.</p> <p>The remaining statements are explained separately.</p>
Required Privilege Level	<p>interface—To view this statement in the configuration.</p> <p>interface-control—To add this statement to the configuration.</p>
Related Topics	<ul style="list-style-type: none"> ■ Configuring DNS and TFTP Packet Forwarding on page 337

export-format

Syntax	export-format cflowd-version-5;
Hierarchy Level	[edit forwarding-options monitoring <i>group-name</i> family inet output]
Release Information	Statement introduced before JUNOS Release 7.4.
Description	Flow monitoring export format.
Options	cflowd-version-5—Cflowd version 5.
Required Privilege Level	<p>interface—To view this statement in the configuration.</p> <p>interface-control—To add this statement to the configuration</p>
Related Topics	<ul style="list-style-type: none"> ■ Configuring Flow Monitoring on page 332

family

See the following sections:

- family (Filtering) on page 362
- family (Monitoring) on page 363
- family (Port Mirroring) on page 364
- family (Sampling) on page 365

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 Release 7.4.
 route-accounting option introduced in JUNOS 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 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 Topics ■ Applying Filters to Forwarding Tables on page 329

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 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 Topics ■ Configuring Flow Monitoring on page 332

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 Release 7.4.
 vpls option introduced in JUNOS Release 9.3 for MX Series routers only; support
 extended to M7i, M10i, M120, and M320 routers in JUNOS Release 9.5.
 ccc option introduced in JUNOS 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 Topics ■ Configuring Port Mirroring on page 341

family (Sampling)

Syntax	family (inet inet6 mpls) { max-packets-per-second <i>number</i> ; maximum-packet-length <i>bytes</i> ; rate <i>number</i> ; run-length <i>number</i> ; }
Hierarchy Level	[edit forwarding-options sampling input]
Release Information	Statement introduced before JUNOS Release 7.4. mpls option introduced in JUNOS Release 8.3.
Description	Configure the protocol family to be sampled.
Options	inet—IP version 4 (IPv4) 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 Topics	■ Configuring Traffic Sampling on page 313

family inet

Syntax	family inet { layer-3; layer-4; }
Hierarchy Level	[edit forwarding-options hash-key]
Release Information	Statement introduced before JUNOS Release 7.4.
Description	Configure layer information for the load-balancing specification. Only the IPv4 protocol is supported.
Options	layer-3—Include Layer 3 (IP) data in the hash key. layer-4—Include Layer 4 Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) data 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 Topics	■ Overview of Per-Packet Load Balancing on page 144

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 Release 7.4.
 no-label-1-exp option introduced in JUNOS Release 8.0.
 label-3 and no-labels options introduced in JUNOS Release 8.1.
 ether-pseudowire statement introduced in JUNOS Release 9.1 (M320 and T Series routers only); support extended to M120 and MX Series routers in JUNOS Release 9.4.

Description For aggregated Ethernet and SONET/SDH interfaces only, configure load balancing based on MPLS labels. Only the IPv4 protocol is supported.

Options

- label-1—Include only one label in the hash key.
- label-2—Include both labels in the hash key.
- label-3—Include the third MPLS label in the hash key.
- no-labels—Include no MPLS labels in the hash key.
- no-label-1-exp—Do not use the EXP bit of the first label in the hash calculation.
- payload—Include bits from 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—Include the IP address of the IPv4 or IPv6 payload in the hash key.
- layer-3-only—Include only Layer 3 IP information.

port-data—Include the source and destination port field information.

source-msb—Include the most significant byte of the source port.

source-lsb—Include the least significant byte of the source port.

destination-msb—Include the most significant byte of the destination port.

destination-lsb—Include the least significant byte of the destination port.

Required Privilege Level

interface—To view this statement in the configuration.

interface-control—To add this statement to the configuration.

Related Topics

- Configuring Load Balancing Based on MPLS Labels on page 147
- Configuring Load Balancing for Ethernet Pseudowires on page 150

family multiservice

Syntax

```
family multiservice {
    destination-mac;
    source-mac;
    label-1;
    label-2;
    payload {
        ip {
            layer-3-only;
            layer-3 {
                (source-address-only | destination-address-only);
            }
            layer-4;
        }
    }
}
```

Hierarchy Level [edit forwarding-options hash-key]

Release Information Statement introduced in JUNOS Release 8.0.
 ip, label-1, label-2, layer-3-only, and payload statements introduced in JUNOS Release 9.4
 layer-3, layer-. source-address-only, and destination-address-only statements introduced in JUNOS Release 9.5.

Description (M Series, MX Series, and T Series routers only) Configure load balancing based on Layer 2 media access control information. On M120 and M320 routers only, configure VPLS load balancing based on MPLS labels and IP information. On MX Series routers, configure VPLS load balancing.

Options

- destination-mac—Include the destination-address MAC information in the hash key.
- source-mac—Include the source-address MAC information in the hash key.
- label-1 (M120 and M320 routers only)—Include the first MPLS label in the hash key.
- label-2 (M120 and M320 routers only)—Include the second MPLS label in the hash key.
- payload (MX Series, M120, and M320 routers only)—Include the packet's IP payload in the hash key
- ip (MX Series, M120, and M320 routers only)—Include the IP address of the IPv4 or IPv6 payload in the hash key.
- layer-3-only (M120, and M320 routers only)—Include only the Layer 3 information from the packets' IP payload in the hash key.
- layer-3 (MX Series routers only)—Include Layer 3 information from the packets' IP payload in the hash key.

source-address-only (MX Series routers only)—Include only the source IP address in the payload in the hash key.

destination-address-only (MX Series routers only)—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.

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

- Related Topics**
- Configuring Load Balancing Based on MAC Addresses on page 151
 - Configuring VPLS Load Balancing Based on IP and MPLS Information on page 151
 - Configuring VPLS Load Balancing on MX Series Ethernet Services Routers on page 153

file

See the following sections:

- file (Extended DHCP Relay Agent and Helpers Trace Options) on page 371
- file (Packet Capture) on page 371
- file (Sampling) on page 372
- file (Trace Options) on page 372

file (Extended DHCP Relay Agent and Helpers Trace Options)

Syntax	file <i>filename</i> <files <i>number</i> > <match <i>regular-expression</i> > <size <i>bytes</i> > <world-readable no-world-readable>;
Hierarchy Level	[edit forwarding-options dhcp-relay traceoptions], [edit forwarding-options helpers traceoptions]
Release Information	Statement introduced before JUNOS Release 7.4.
Description	Configure information about the DNS and TFTP packet-forwarding files that contain trace logging information.
Options	<i>filename</i> —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 Topics	■ Tracing BOOTP, DNS, and TFTP Forwarding Operations on page 338

file (Packet Capture)

Syntax	file filename <i>filename</i> <files <i>number</i> > <size <i>bytes</i> > <world-readable no-world-readable>;
Hierarchy Level	[edit forwarding-options packet-capture]
Release Information	Statement introduced in JUNOS 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 Topics	■ Configuring Packet Capture on page 345

file (Sampling)

Syntax	file filename <i>filename</i> <disable> <files <i>number</i> > <stamp no-stamp> <size <i>bytes</i> > <world-readable no-world-readable>;
Hierarchy Level	[edit forwarding-options sampling output]
Release Information	Statement introduced before JUNOS 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 Topics	■ Configuring the Output File for Traffic Sampling on page 315

file (Trace Options)

Syntax	file <i>filename</i> <files <i>number</i> > <size <i>bytes</i> > <world-readable no-world-readable>;
Hierarchy Level	[edit forwarding-options port-mirroring traceoptions], [edit forwarding-options sampling traceoptions]
Release Information	Statement introduced before JUNOS Release 7.4.
Description	Configure information about the files that contain trace logging information.
Options	<i>filename</i> —The name of the file containing the trace information. Default: /var/log/sampled The remaining statements are explained separately.
Usage Guidelines	See “Tracing Traffic Sampling Operations” on page 317.
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.

filename

See the following sections:

- filename (Packet Capture) on page 373
- filename (Sampling) on page 373

filename (Packet Capture)

Syntax	filename <i>filename</i> ;
Hierarchy Level	[edit forwarding-options packet-capture file]
Release Information	Statement introduced in JUNOS Release 7.5.
Description	Configure the name of the output file.
Options	<i>filename</i> —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 Topics	■ Configuring Packet Capture on page 345

filename (Sampling)

Syntax	filename <i>filename</i> ;
Hierarchy Level	[edit forwarding-options sampling output file]
Release Information	Statement introduced before JUNOS Release 7.4.
Description	Configure the name of the output file.
Options	<i>filename</i> —Name of the file in which to place the traffic samples. All files are placed in the directory <code>/var/tmp</code> .
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.
Related Topics	■ Configuring the Output File for Traffic Sampling on page 315

files

See the following sections:

- files (Packet Capture) on page 374
- files (Sampling and Traceoptions) on page 374

files (Packet Capture)

Syntax	files <i>number</i> ;
Hierarchy Level	[edit forwarding-options packet-capture file]
Release Information	Statement introduced in JUNOS Release 7.5.
Description	Configure the maximum number of files for packet capturing.
Options	<i>number</i> —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 Topics	<ul style="list-style-type: none"> ■ Configuring Packet Capture on page 345

files (Sampling and Traceoptions)

Syntax	files <i>number</i> ;
Hierarchy Level	[edit forwarding-options helpers traceoptions file], [edit forwarding-options port-mirroring traceoptions file], [edit forwarding-options sampling output file], [edit forwarding-options sampling traceoptions file]
Release Information	Statement introduced before JUNOS Release 7.4.
Description	Configure the total number of files to be saved with samples or trace data.
Options	<i>number</i> —Maximum number of traffic sampling or trace log files. When a file named <i>sampling-file</i> reaches its maximum size, it is renamed <i>sampling-file.0</i> , then <i>sampling-file.1</i> , 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 Topics	<ul style="list-style-type: none"> ■ Configuring the Output File for Traffic Sampling on page 315 ■ Tracing Traffic Sampling Operations on page 317

filter

See the following sections:

- filter (IPv4, IPv6, and MPLS) on page 375
- filter (VPLS) on page 375

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 Release 7.4.

Description Apply a forwarding table filter to a forwarding table.

Options The statements are explained separately.

Usage Guidelines See “Applying Filters to Forwarding Tables” on page 329.

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

filter (VPLS)

Syntax filter input *filter-name*;

Hierarchy Level [edit forwarding-options family vpls]

Release Information Statement introduced before JUNOS Release 7.4.

Description Apply a forwarding table filter for VPLS.

Options The other statement is explained separately.

Usage Guidelines See “Applying Filters to Forwarding Tables” on page 329.

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

flood

Syntax	flood { input <i>filter-name</i> ; }
Hierarchy Level	[edit forwarding-options family vpls]
Release Information	Statement introduced before JUNOS Release 7.4.
Description	Apply a forwarding table filter to a flood table.
Options	input <i>filter-name</i> —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 Topics	■ Applying Filters to Forwarding Tables on page 329

flow-active-timeout

Syntax	flow-active-timeout <i>seconds</i> ;
Hierarchy Level	[edit forwarding-options accounting <i>group-name</i> output], [edit forwarding-options monitoring <i>group-name</i> family inetoutput], [edit forwarding-options sampling output]
Release Information	Statement introduced before JUNOS Release 7.4. Statement introduced in JUNOS Release 9.0 for EX Series switches.
Description	Configure the interval before exporting an active flow.
Options	<i>seconds</i> —Interval, 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 Topics	■ Configuring Discard Accounting on page 330 ■ Configuring Flow Monitoring on page 332 ■ Configuring the Output File for Traffic Sampling on page 315

flow-export-destination

Syntax	flow-export-destination { (cflowd-collector collector-pic); }
Hierarchy Level	[edit forwarding-options monitoring <i>group-name</i> family inet output]
Release Information	Statement introduced before JUNOS 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 Topics	■ Configuring Flow Monitoring on page 332

flow-inactive-timeout

Syntax	flow-inactive-timeout <i>seconds</i> ;
Hierarchy Level	[edit forwarding-options accounting <i>group-name</i> output], [edit forwarding-options monitoring <i>group-name</i> family inet output], [edit forwarding-options sampling output]
Release Information	Statement introduced before JUNOS Release 7.4. Statement introduced in JUNOS Release 9.0 for EX Series switches.
Description	Configure the interval before a flow is considered inactive.
Options	<i>seconds</i> —Interval, 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 Topics	■ Configuring Discard Accounting on page 330 ■ Configuring Flow Monitoring on page 332 ■ Configuring the Output File for Traffic Sampling on page 315

forwarding-options

Syntax	forwarding-options { ... }
Hierarchy Level	[edit]
Release Information	Statement introduced before JUNOS 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.
Related Topics	■ Introduction to Traffic Sampling Configuration on page 311

group (DHCP Spoofing Prevention)

Syntax	group <i>group-name</i> { interface <i>interface-name</i> ; }
Hierarchy Level	[edit routing-instances <i>routing-instance-name</i> bridge-domains <i>bridge-domain-name</i> forwarding-options dhcp-relay], [edit routing-instances <i>routing-instance-name</i> forwarding-options dhcp-relay]
Release Information	Statement introduced in JUNOS 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.
Usage Guidelines	See “Preventing DHCP Spoofing on MX Series Ethernet Services Routers” on page 340.
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.

hash-key

Syntax

```

hash-key {
  family inet {
    layer-3;
    layer-4;
  }
  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 Release 7.4.
 family multiservice and no-label-1-exp options introduced in JUNOS Release 8.0.
 label-3 and no-labels options introduced in JUNOS Release 8.1.
 ether-pseudowire statement introduced in JUNOS Release 9.1 (M320 and T Series routers only); support extended to M120 and MX Series routers in JUNOS Release 9.4.

`ip`, `label-1`, `label-2`, `layer-3-only`, and `payload` options for the `family multiservice` statement introduced in JUNOS 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 Release 9.5. (MX Series routers only)

Description Select which packet header data to use for per-flow load balancing.

Options `inet`—IP address family.

`mpls`—MPLS address family.

`multiservice`—Multiservice protocol family

`layer-3`—Incorporate Layer 3 data into the hash key.

`layer-4`—Incorporate Layer 4 data into the hash key.

`no-label-1-exp`—The EXP bit of the first label is not used in the hash calculation.

`label-1`—Incorporate the first label into the hash key.

`label-2`—Incorporate the second label into the hash key.

`label-3`—Include the third MPLS label in the hash key.

`no-labels`—Include no MPLS labels in the hash key.

`payload`—Incorporate payload data into the hash key.

`ip`—Include the IP address of the IPv4 or IPv6 payload in the hash key.

`layer-3-only`—Include only Layer 3 IP information.

`port-data`—Include the source and destination port field information.

`source-msb`—Include the most significant byte of the source port.

`source-lsb`—Include the least significant byte of the source port.

`destination-msb`—Include the most significant byte of the destination port.

`destination-lsb`—Include the least significant byte of the destination port.

`destination-mac`—Include the destination MAC address in the hash key.

`source-address-only`—Include only the Layer 3 IP source address in the hash key.

`destination-address-only`—Include only the Layer 3 IP destination address 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 Topics ■ Overview of Per-Packet Load Balancing on page 144

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 [ routing-instance-names ]>;
                    }
                }
                maximum-hop-count number;
                minimum-wait-time seconds;
                server address {
                    <logical-system logical-system-name>
                    <routing-instance [ 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 Release 7.4.
Statement introduced in JUNOS 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 Topics

- Configuring DNS and TFTP Packet Forwarding on page 337
- Configuring Routers, Switches and Interfaces as DHCP and BOOTP Relay Agents on page 335

indexed-next-hop

Syntax	indexed-next-hop;
Hierarchy Level	[edit forwarding-options load-balance], [edit logical-systems <i>logical-system-name</i> forwarding-options load-balance], [edit logical-systems <i>logical-system-name</i> routing-instances <i>routing-instance-name</i> forwarding-options load-balance], [edit routing-instances <i>routing-instance-name</i> forwarding-options load-balance]
Release Information	Statement introduced in JUNOS Release 9.0. Statement introduced in JUNOS Release 9.0 for EX Series switches.
Description	Generate a permuted index of next-hop entries for unicast and aggregate next hops.
Required Privilege Level	routing—To view this statement in the configuration. routing-control—To add this statement to the configuration.
Related Topics	■ Configuring Per-Prefix Load Balancing on page 334

input

See the following sections:

- `input` (Forwarding Table) on page 386
- `input` (Port Mirroring) on page 386
- `input` (Sampling) on page 387

input (Forwarding Table)

Syntax	<code>input filter-name;</code>
Hierarchy Level	[edit forwarding-options family (inet inet6 mpls vpls) filter], [edit routing-instances <i>routing-instance-name</i> forwarding-options family (inet inet6 mpls vpls) filter]
Release Information	Statement introduced before JUNOS Release 7.4.
Description	Apply a forwarding table filter to ingress traffic of the forwarding table.
Options	<i>filter-name</i> —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 Topics	■ Applying Filters to Forwarding Tables on page 329

input (Port Mirroring)

Syntax	<code>input { maximum-packet-length bytes; rate number; run-length number; }</code>
Hierarchy Level	[edit forwarding-options port-mirroring], [edit forwarding-options port-mirroring instance <i>instance-name</i>]
Release Information	Statement introduced before JUNOS Release 7.4. maximum-packet-length option introduced in JUNOS 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 Topics	■ Configuring Port Mirroring on page 341

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 Release 7.4.
 mpls option introduced in JUNOS 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 Topics ■ Configuring Traffic Sampling on page 313

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 Release 9.3 (MX Series routers only). Support extended to M120 and M320 routers in JUNOS Release 9.5. `maximum-packet-length` and `ccc` options introduced in JUNOS 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 Topics ■ Configuring Port Mirroring on page 341

interface

See the following sections:

- interface (Accounting or Sampling) on page 389
- interface (BOOTP) on page 390
- interface (DHCP Spoofing Prevention) on page 391
- interface (DNS and TFTP Packet Forwarding or Relay Agent) on page 391
- interface (Monitoring) on page 392
- interface (Next-Hop Group) on page 392
- interface (Port Mirroring) on page 393

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 output]

Release Information Statement introduced before JUNOS Release 7.4.
 Statement introduced in JUNOS 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 Topics ■ Configuring Discard Accounting on page 330
 ■ Configuring the Output File for Traffic Sampling on page 315

interface (BOOTP)

Syntax 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 [*routing-instance-names*]>;
 }
 }

Hierarchy Level [edit forwarding-options helpers bootp]

Release Information Statement introduced before JUNOS Release 7.4.
 Statement introduced in JUNOS Release 9.0 for EX 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 Topics**
- Configuring Routers, Switches and Interfaces as DHCP and BOOTP Relay Agents on page 335
 - Setting Up DHCP Option 82 with the Switch as a Relay Agent Between Clients and DHCP Server (CLI Procedure)

interface (DHCP Spoofing Prevention)

Syntax	<code>interface interface-name;</code>
Hierarchy Level	[edit routing-instances <i>routing-instance-name</i> bridge-domains <i>bridge-domain-name</i> forwarding-options dhcp-relay group <i>group-name</i> interface <i>interface-name</i>], [edit routing-instances <i>routing-instance-name</i> forwarding-options dhcp-relay group <i>group-name</i> interface <i>interface-name</i>]
Release Information	Statement introduced in JUNOS 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 Topics	■ Preventing DHCP Spoofing on MX Series Ethernet Services Routers on page 340

interface (DNS and TFTP Packet Forwarding or Relay Agent)

Syntax	<code>interface interface-name { broadcast; description text-description; no-listen; server address <logical-system logical-system-name> <routing-instance routing-instance-name>; }</code>
Hierarchy Level	[edit forwarding-options helpers domain], [edit forwarding-options helpers tftp]
Release Information	Statement introduced before JUNOS Release 7.4. Statement introduced in JUNOS Release 9.0 for EX Series switches.
Description	Specify the interface for monitoring and forwarding DNS or TFTP requests.
Options	<i>interface-name</i> —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 Topics	■ Configuring DNS and TFTP Packet Forwarding on page 337

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* family inet output]

Release Information Statement introduced before JUNOS 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 Topics ■ Configuring Flow Monitoring on page 332

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 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 Topics ■ Configuring Next-Hop Groups on page 333

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 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 Topics ■ Configuring Port Mirroring on page 341

load-balance

Syntax	<pre>load-balance { indexed-next-hop; per-flow { hash-seed; } per-prefix { hash-seed <i>number</i>; } }</pre>
Hierarchy Level	[edit forwarding-options], [edit logical-systems <i>logical-system-name</i> forwarding-options], [edit logical-systems <i>logical-system-name</i> routing-instances <i>routing-instance-name</i> forwarding-options], [edit routing-instances <i>routing-instance-name</i> forwarding-options]
Release Information	Statement introduced in JUNOS Release 9.0. Statement introduced in JUNOS Release 9.0 for EX Series switches. Support for per-flow load balancing introduced in JUNOS 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.
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 Topics	<ul style="list-style-type: none"> ■ Configuring Per-Flow Load Balancing Based on Hash Values on page 335 ■ Configuring Per-Prefix Load Balancing on page 334

local-dump

Syntax	(local-dump no-local-dump);
Hierarchy Level	[edit forwarding-options sampling output cflowd <i>hostname</i>]
Release Information	Statement introduced before JUNOS 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 Topics	■ Debugging cflowd Flow Aggregation on page 319

max-packets-per-second

Syntax	max-packets-per-second <i>number</i> ;
Hierarchy Level	[edit forwarding-options sampling input family]
Release Information	Statement introduced before JUNOS 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	<i>number</i> —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 Topics	■ See Configuring Traffic Sampling on page 313.

maximum-capture-size

Syntax	maximum-capture-size <i>bytes</i> ;
Hierarchy Level	[edit forwarding-options packet-capture]
Release Information	Statement introduced in JUNOS Release 7.5.
Description	Configure the maximum size of capture for packets.
Options	<i>bytes</i> —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 Topics	■ Configuring Packet Capture on page 345

maximum-hop-count

Syntax	maximum-hop-count <i>number</i> ;
Hierarchy Level	[edit forwarding-options helpers bootp], [edit forwarding-options helpers bootp interface <i>interface-group</i>]
Release Information	Statement introduced before JUNOS Release 7.4. Statement introduced in JUNOS Release 9.0 for EX Series switches.
Description	Specify the maximum number of hops allowed.
Options	<i>number</i> —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 Topics	■ Configuring Routers, Switches and Interfaces as DHCP and BOOTP Relay Agents on page 335

maximum-packet-length

Syntax	maximum-packet-length <i>bytes</i> ;
Hierarchy Level	[edit forwarding-options port-mirroring input], [edit forwarding-options port-mirroring instance <i>instance-name</i> input],
Release Information	Statement introduced in JUNOS Release 9.6.
Description	Set the maximum length of the packet used for port mirroring. Packets with lengths greater than the specified maximum are truncated.
Options	<i>bytes</i> —Number of bytes.
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.
Related Topics	■ Configuring Port Mirroring on page 341

minimum-wait-time

Syntax	minimum-wait-time <i>seconds</i> ;
Hierarchy Level	[edit forwarding-options helpers bootp], [edit forwarding-options helpers bootp interface <i>interface-group</i>]
Release Information	Statement introduced before JUNOS Release 7.4. Statement introduced in JUNOS Release 9.0 for EX Series switches.
Description	Specify the minimum time allowed.
Options	<i>seconds</i> —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 Topics	■ Configuring Routers, Switches and Interfaces as DHCP and BOOTP Relay Agents on page 335

mirror-once

Syntax	mirror-once;
Hierarchy Level	[edit forwarding-options port-mirroring]
Release Information	Statement introduced in JUNOS Release 9.3 (MX Series routers only). Support extended to M120 routers in JUNOS 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 Topics	■ Configuring Port Mirroring on page 341

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 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 Topics ■ Configuring Flow Monitoring on page 332

next-hop

Syntax	next-hop <i>address</i> ;
Hierarchy Level	[edit forwarding-options port-mirroring output interface <i>interface-name</i>], [edit forwarding-options port-mirroring family (inet inet6) output interface <i>interface-name</i>]
Release Information	Statement introduced before JUNOS Release 7.4.
Description	Specify the next-hop address for sending copies of packets to an analyzer.
Options	<i>address</i> —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 Topics	■ Configuring Port Mirroring on page 341

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 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 Topics ■ Configuring Next-Hop Groups on page 333

no-filter-check

Syntax	no-filter-check;
Hierarchy Level	[edit forwarding-options port-mirroring output], [edit forwarding-options port-mirroring family (inet inet6) output]
Release Information	Statement introduced before JUNOS 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 Topics	■ Configuring Port Mirroring on page 341

no-listen

Syntax	no-listen;
Hierarchy Level	[edit forwarding-options helpers bootp interface <i>interface-group</i>], [edit forwarding-options helpers domain interface <i>interface-name</i>], [edit forwarding-options helpers tftp interface <i>interface-name</i>]
Release Information	Statement introduced before JUNOS Release 7.4. Statement introduced in JUNOS Release 9.0 for EX 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 Topics	■ Configuring DNS and TFTP Packet Forwarding on page 337 ■ Configuring Routers, Switches and Interfaces as DHCP and BOOTP Relay Agents on page 335

no-local-dump

See local-dump

no-stamp

See stamp

no-world-readable

See world-readable

output

See the following sections:

- output (Accounting) on page 405
- output (Forwarding Table) on page 406
- output (Monitoring) on page 406
- output (Port Mirroring) on page 407
- output (Sampling) on page 408

output (Accounting)

Syntax	<pre>output { cflowd [<i>hostnames</i>] { aggregation { autonomous-system; destination-prefix; protocol-port; source-destination-prefix { caida-compliant; } source-prefix; } autonomous-system-type (origin peer); port <i>port-number</i>; version <i>format</i>; } flow-active-timeout <i>seconds</i>; flow-inactive-timeout <i>seconds</i>; interface <i>interface-name</i> { engine-id <i>number</i>; engine-type <i>number</i>; source-address <i>address</i>; } }</pre>
Hierarchy Level	[edit forwarding-options accounting <i>group-name</i>]
Release Information	Statement introduced before JUNOS Release 7.4.
Description	<p>Configure cflowd, output interfaces, and flow properties.</p> <p>The statements are explained separately.</p>
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.
Related Topics	■ Configuring Discard Accounting on page 330

output (Forwarding Table)

Syntax	<code>output filter-name;</code>
Hierarchy Level	[edit forwarding-options family (inet inet6 mpls) filter], [edit routing-instances <i>routing-instance-name</i> forwarding-options family (inet inet6 mpls) filter]
Release Information	Statement introduced in JUNOS Release 7.5.
Description	Configure filtering on the egress traffic of the forwarding table.
Options	<i>filter-name</i> —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 Topics	■ Applying Filters to Forwarding Tables on page 329

output (Monitoring)

Syntax	<pre>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; } }</pre>
Hierarchy Level	[edit forwarding-options monitoring <i>group-name</i> family inet]
Release Information	Statement introduced before JUNOS 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 Topics	■ Configuring Flow Monitoring on page 332

output (Port Mirroring)

Syntax	<pre>output { interface <i>interface-name</i> { next-hop <i>address</i>; } no-filter-check; }</pre>
Hierarchy Level	[edit forwarding-options port-mirroring family (ccc inet inet6 vpls)], [edit forwarding-options port-mirroring instance <i>instance-name</i> family (ccc inet inet6 vpls)]
Release Information	Statement introduced before JUNOS Release 7.4. vpls option introduced in JUNOS Release 9.3 for MX Series routers only; support extended to M7i, M10i, M120, and M320 routers in JUNOS Release 9.5. ccc option introduced in JUNOS 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 Topics	■ Configuring Port Mirroring on page 341

output (Sampling)

Syntax

```

output {
  aggregate-export-interval seconds;
  cflowd 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 Release 7.4.
version9 statement introduced in JUNOS 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 Topics**
- Configuring Active Flow Monitoring Using Version 9 on page 320
 - Configuring the Output File for Traffic Sampling on page 315

packet-capture

Syntax	<pre>packet-capture { disable; file filename <i>filename</i> <files <i>number</i>> <size <i>bytes</i>> <world-readable no-world-readable>; maximum-capture-size <i>number</i>; }</pre>
Hierarchy Level	[edit forwarding-options]
Release Information	Statement introduced in JUNOS Release 7.5.
Description	Configure packet capture on a router.
Options	<p>disable—Disable packet capture on the router.</p> <p>The remaining statements are explained separately.</p>
Required Privilege Level	<p>interface—To view this statement in the configuration.</p> <p>interface-control—To add this statement to the configuration.</p>
Related Topics	<ul style="list-style-type: none"> ■ Configuring Packet Capture on page 345

per-flow

Syntax	<pre>per-flow { hash-seed; }</pre>
Hierarchy Level	<p>[edit forwarding-options load-balance],</p> <p>[edit logical-systems <i>logical-system-name</i> forwarding-options load-balance],</p> <p>[edit logical-systems <i>logical-system-name</i> routing-instances <i>routing-instance-name</i> forwarding-options load-balance],</p> <p>[edit routing-instances <i>routing-instance-name</i> forwarding-options load-balance]</p>
Release Information	Statement introduced in JUNOS 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 Software automatically chooses a value for the hashing algorithm used. You cannot configure a specific hash value for per-flow load balancing.
Required Privilege Level	<p>routing—To view this statement in the configuration.</p> <p>routing-control—To add this statement to the configuration.</p>
Related Topics	<ul style="list-style-type: none"> ■ Configuring Per-Flow Load Balancing Based on Hash Values on page 335 ■ load-balance

per-prefix

Syntax	per-prefix { hash-seed <i>number</i> ; }
Hierarchy Level	[edit forwarding-options load-balance], [edit logical-systems <i>logical-system-name</i> forwarding-options load-balance], [edit logical-systems <i>logical-system-name</i> routing-instances <i>routing-instance-name</i> forwarding-options load-balance], [edit routing-instances <i>routing-instance-name</i> forwarding-options load-balance]
Release Information	Statement introduced in JUNOS Release 9.0. Statement introduced in JUNOS 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. <i>number</i> —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 Topics	■ load-balance ■ Configuring Per-Prefix Load Balancing on page 334

port

Syntax	<code>port <i>port-number</i>;</code>
Hierarchy Level	[edit forwarding-options accounting <i>group-name</i> output cflowd <i>hostname</i>], [edit forwarding-options monitoring <i>group-name</i> family inet output cflowd <i>hostname</i>], [edit forwarding-options sampling output cflowd <i>hostname</i>]
Release Information	Statement introduced before JUNOS Release 7.4.
Description	Specify the UDP port number on the cflowd host system.
Options	<i>port-number</i> —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 Topics	■ Configuring Flow Aggregation (cflowd) on page 317

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 Release 7.4.
family vpls statement introduced in JUNOS Release 9.3 (MX Series routers only); support extended to M7i, M10, M120, and M320 routers in JUNOS Release 9.5.
instance *port-mirroring-instance-name* statement introduced in JUNOS Release 9.3 (MX Series routers only); support extended to M120 and M320 routers in JUNOS Release 9.5.
mirror-once statement introduced in JUNOS Release 9.3 (MX Series routers only); support extended to M120 routers in JUNOS Release 9.5.
family ccc statement introduced in JUNOS 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.



NOTE: For information about configuring port mirroring for Layer 2 traffic on MX Series routers, see the *JUNOS MX Series Ethernet Services Routers Solutions Guide*.

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

Related Topics ■ Configuring Port Mirroring on page 341

rate

Syntax rate *number*;

Hierarchy Level [edit forwarding-options port-mirroring input],
[edit forwarding-options port-mirroring instance *instance-name* input],
[edit forwarding-options sampling input family]

Release Information Statement introduced before JUNOS 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 Topics ■ Configuring Port Mirroring on page 341
■ Configuring Traffic Sampling on page 313

route-accounting

Syntax	route-accounting;
Hierarchy Level	[edit forwarding-options family inet6]
Release Information	Statement introduced in JUNOS 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 Topics	■ Configuring IPv6 Accounting on page 330

run-length

Syntax	run-length <i>number</i> ;
Hierarchy Level	[edit forwarding-options port-mirroring input], [edit forwarding-options port-mirroring instance <i>port-mirroring-instance-name</i> input], [edit forwarding-options sampling input family (inet inet6 mpls)]
Release Information	Statement introduced before JUNOS 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.
Options	<i>number</i> —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 Topics	■ Applying Filters to Forwarding Tables on page 329

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;
                cflowd 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;
        }
    }

```



```
    }
    traceoptions {
      file filename <files number> <size bytes> <world-readable | no-world-readable>;
    }
  }
```

Hierarchy Level [edit forwarding-options]

Release Information Statement introduced before JUNOS 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 Topics**
- Applying Filters to Forwarding Tables on page 329
 - Configuring Active Flow Monitoring Using Version 9 on page 320
 - Configuring Flow Aggregation (cflowd) on page 317
 - Configuring Port Mirroring on page 341
 - Tracing Traffic Sampling Operations on page 317

server

See the following sections:

- server (DHCP and BOOTP Relay Agent) on page 419
- server (DNS and TFTP Service) on page 419

server (DHCP and BOOTP Relay Agent)

Syntax	server address { <logical-system <i>logical-system-name</i> > <routing-instance [<i>routing-instance-names</i>]>; }
Hierarchy Level	[edit forwarding-options helpers bootp], [edit forwarding-options helpers bootp interface <i>interface-group</i>]
Release Information	Statement introduced before JUNOS Release 7.4. Statement introduced in JUNOS Release 9.0 for EX Series switches.
Description	Configure the router or switch to act as a DHCP and BOOTP relay agent.
Options	<ul style="list-style-type: none"> ■ <i>address</i>—One or more addresses of the server. ■ <i>logical-system</i>—Logical system of server.
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.
Related Topics	<ul style="list-style-type: none"> ■ Configuring Routers, Switches and Interfaces as DHCP and BOOTP Relay Agents on page 335

server (DNS and TFTP Service)

Syntax	server address <logical-system <i>logical-system-name</i> > <routing-instance <i>routing-instance-name</i> >;
Hierarchy Level	[edit forwarding-options helpers domain], [edit forwarding-options helpers domain interface <i>interface-name</i>], [edit forwarding-options helpers tftp], [edit forwarding-options helpers tftp interface <i>interface-name</i>]
Release Information	Statement introduced before JUNOS Release 7.4. Statement introduced in JUNOS 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	<i>address</i> —Address of the server. <i>routing-instance</i> [<i>routing-instance-names</i>]—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 Topics	<ul style="list-style-type: none"> ■ Configuring DNS and TFTP Packet Forwarding on page 337

size

See the following sections:

- size (Packet Capture) on page 421
- size (Sampling and Traceoptions) on page 421

size (Packet Capture)

Syntax	<code>size number;</code>
Hierarchy Level	[edit forwarding-options packet-capture file]
Release Information	Statement introduced in JUNOS Release 7.5.
Description	Configure the maximum size of the file for packet capturing.
Options	<i>number</i> —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 Topics	<ul style="list-style-type: none"> ■ Configuring Packet Capture on page 345

size (Sampling and Traceoptions)

Syntax	<code>size bytes;</code>
Hierarchy Level	[edit forwarding-options helpers traceoptions file], [edit forwarding-options port-mirroring traceoptions file], [edit forwarding-options sampling output file], [edit forwarding-options sampling traceoptions file]
Release Information	Statement introduced before JUNOS Release 7.4.
Description	<p>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.</p> <p>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.</p>
Options	<i>bytes</i> —Maximum size of each traffic sampling file or trace log file, in kilobytes (KB), megabytes (MB), or gigabytes (GB). Syntax: <i>xk</i> to specify KB, <i>xm</i> to specify MB, or <i>xg</i> 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 Topics	<ul style="list-style-type: none"> ■ Configuring the Output File for Traffic Sampling on page 315 ■ Tracing Traffic Sampling Operations on page 317

stamp

Syntax	(stamp no-stamp);
Hierarchy Level	[edit forwarding-options sampling output file]
Release Information	Statement introduced before JUNOS Release 7.4.
Description	Include a timestamp with each line in the output file.
Options	<p>no-stamp—Do not include timestamps. This is the default.</p> <p>stamp—Include a timestamp with each line of packet sampling information.</p> <p>Default: No timestamp is included.</p>
Required Privilege Level	<p>interface—To view this statement in the configuration.</p> <p>interface-control—To add this statement to the configuration.</p>
Related Topics	■ Configuring the Output File for Traffic Sampling on page 315

tftp

Syntax	<pre>tftp { description <i>text-description</i>; interface <i>interface-name</i> { broadcast; description <i>text-description</i>; no-listen; server <i>address</i> <logical-system <i>logical-system-name</i>> <routing-instance <i>routing-instance-name</i>>; } server <i>address</i> <logical-system <i>logical-system-name</i>> <routing-instance <i>routing-instance-name</i>>; }</pre>
Hierarchy Level	[edit forwarding-options helpers]
Release Information	<p>Statement introduced before JUNOS Release 7.4.</p> <p>Statement introduced in JUNOS Release 9.0 for EX Series switches.</p>
Description	<p>Enable TFTP request packet forwarding.</p> <p>The remaining statements are explained separately.</p>
Required Privilege Level	<p>interface—To view this statement in the configuration.</p> <p>interface-control—To add this statement to the configuration.</p>
Related Topics	■ Configuring DNS and TFTP Packet Forwarding on page 337

traceoptions

See the following sections:

- **traceoptions** (DNS and TFTP Packet Forwarding) on page 424
- **traceoptions** (Port Mirroring and Traffic Sampling) on page 426

traceoptions (DNS and TFTP Packet Forwarding)

Syntax traceoptions {
 file *filename* <files (Sampling and Traceoptions) *number*> <match *regular-expression*>
 <size (Sampling and Traceoptions) *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 Release 7.4.
 Statement standardized and **match** option introduced in JUNOS Release 8.0.
 Statement introduced in JUNOS 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.

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: **xk** to specify KB, **xm** to specify MB, or **xg** 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 Topics ■ Tracing BOOTP, DNS, and TFTP Forwarding Operations on page 338

traceoptions (Port Mirroring and Traffic Sampling)

Syntax	traceoptions { file <i>filename</i> <files <i>number</i> > <size <i>bytes</i> > <world-readable no-world-readable>; }
Hierarchy Level	[edit forwarding-options port-mirroring], [edit forwarding-options sampling]
Release Information	Statement introduced before JUNOS 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 Topics	■ Tracing Traffic Sampling Operations on page 317

version

Syntax	version <i>format</i> ;
Hierarchy Level	[edit forwarding-options accounting <i>group-name</i> output cflowd <i>hostname</i>], [edit forwarding-options sampling output cflowd <i>hostname</i>]
Release Information	Statement introduced before JUNOS Release 7.4.
Description	Specify the version format of the aggregated flows exported to a cflowd server.
Options	<i>format</i> —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 Topics	■ Configuring Flow Aggregation (cflowd) on page 317

version9

Syntax	version9 { template <i>template-name</i> ; }
Hierarchy Level	[edit forwarding-options sampling cflowd <i>hostname</i>]
Release Information	Statement introduced in JUNOS Release 8.3.
Description	Enable active flow monitoring using the version 9 template format to collect traffic flows.
Options	<i>template-name</i> —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 Topics	<ul style="list-style-type: none"> ■ Configuring Active Flow Monitoring Using Version 9 on page 320 ■ <i>JUNOS Services Interfaces Configuration Guide</i> and <i>JUNOS Feature Guide</i>

world-readable

Syntax	(world-readable no-world-readable);
Hierarchy Level	[edit forwarding-options helpers traceoptions file], [edit forwarding-options packet-capture file], [edit forwarding-options port-mirroring traceoptions file], [edit forwarding-options sampling output file], [edit forwarding-options sampling traceoptions file]
Release Information	Statement introduced before JUNOS Release 7.4. Statement introduced in JUNOS Release 9.0 for EX Series switches.
Description	Enable unrestricted file access.
Options	no-world-readable—Restrict file access to owner. world-readable—Enable unrestricted file access. Default: no-world-readable
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.
Related Topics	<ul style="list-style-type: none"> ■ Configuring the Output File for Traffic Sampling on page 315 ■ Tracing Traffic Sampling Operations on page 317

Part 5

Indexes

- Index on page 431
- Index of Statements and Commands on page 439

Index

Symbols

!	
in policy expressions	
logical operator.....	61
! (negation)	
in firewall filters	
bit-field logical operator.....	207
#, comments in configuration statements.....	xxx
&&, logical operator.....	60
&, bit-field logical operator.....	207
(), in syntax descriptions.....	xxx
+	
bit-field logical operator.....	207
, (comma), bit-field logical operator.....	207
< >, in syntax descriptions.....	xxix
[], in configuration statements.....	xxx
{ }, in configuration statements.....	xxx
(pipe)	
in firewall filters	
bit-field logical operator.....	207
(pipe), in syntax descriptions.....	xxx
(pipes), logical operator.....	60

A

accept	
firewall filter action.....	208
firewall filters	
action.....	312
policy, routing	
control action.....	49
access and access-internal routes	
importing and exporting in routing policies.....	74
accounting statement.....	350
usage guidelines.....	330
accounting-profile statement.....	287
firewall	
usage guidelines.....	247
action modifiers, firewall filters.....	208
action statement.....	288
policer (TCM)	
usage guidelines.....	280

actions

firewall filters.....	174, 208
policy, routing.....	25
characteristics, manipulating.....	49
flow control.....	47, 49
tracing.....	47
tracing.....	56
address (filter match conditions).....	203
aggregate policer.....	271
aggregate statement	
usage guidelines.....	272
aggregation statement.....	351
usage guidelines.....	330
ampersand (&), bit-field logical operator.....	207
apply-path statement.....	155
usage guidelines.....	118
area (routing policy match condition).....	43
as-path (routing policy match condition).....	43
as-path statement.....	156
policy, routing	
usage guidelines.....	98
as-path-group statement.....	157
usage guidelines.....	98
as-path-prepend (routing policy action).....	49, 137
ASs	

paths

modifying with routing policy.....	49, 137
regular expressions <i>See</i> policy, routing, AS path	
regular expressions	
autonomous-system-type statement.....	352
usage guidelines.....	317

B

bandwidth policer.....	277
BGP	
communities	
names.....	106
policy, routing.....	104, 158
damping parameters.....	138, 161
dynamic routing policies	
applying.....	68
extended communities.....	109
policy, routing	
applying.....	28

bit-field	
firewall filter match conditions.....	206
logical operators.....	207
BOOTP relay agent.....	335
bootp statement.....	353
usage guidelines.....	335
braces, in configuration statements.....	xxx
brackets	
angle, in syntax descriptions.....	xxix
square, in configuration statements.....	xxx

C

cflowd statement.....	354
usage guidelines.....	317
class (routing policy action).....	50
client-response-ttl statement.....	358
color	
policy, routing	
action.....	50
match condition.....	43
comments, in configuration statements.....	xxx
communities	
extend range of BGP communities.....	109
names.....	106
policy, routing.....	104, 158
action.....	50
match condition.....	43
community statement.....	158
policy, routing	
usage guidelines.....	106
condition statement.....	160
conventions	
text and syntax.....	xxix
count	
firewall filter action modifier.....	208
curly braces, in configuration statements.....	xxx
customer support.....	xxx
contacting JTAC.....	xxx

D

damping	
policy, routing, action.....	50
damping statement.....	161
BGP	
usage guidelines.....	142
policy, routing	
usage guidelines.....	139
description statement	
helper service or interface.....	358
service	
usage guidelines.....	335, 337
destination class usage.....	51, 52
destination-class (routing policy action).....	51, 54
destination-port (firewall filter match condition).....	200

DHCP

relay agent.....	335
relay agents, extended	
access and access-internal routes.....	74
snooping.....	340
dhcp-relay statement	
DHCP snooping.....	359
disable statement	
packet capture.....	359
sampling.....	359
traffic sampling	
usage guidelines.....	315
discard	
firewall filter action.....	208
discard interface.....	71
DNS	
packet forwarding.....	337
requests, disabling recognition.....	337
documentation	
comments on.....	xxx
domain statement.....	360
dscp (firewall filter match condition).....	187
DVMRP	
policy, routing	
applying.....	28
dynamic database	
active nonstop routing.....	68
routing policies.....	66
dynamic routing policies	
active nonstop routing.....	68
BGP.....	68
configuring.....	67
dynamic-db statement.....	162
overview.....	65
dynamic-db statement.....	162
usage guidelines.....	67

E

ether-pseudowire statement.....	366
usage guidelines.....	150
evaluation	
firewall filters.....	182
policy, routing	29
exact route list match type.....	120, 123
exclamation point (!), bit-field logical operator.....	207
export routing policies	
applying.....	27, 57, 163
from statement.....	59
export statement.....	163
policy, routing	
usage guidelines.....	27, 57, 64
export-format statement.....	360
usage guidelines.....	332

F

- family inet statement (firewall filter)
 - usage guidelines.....178
- family inet statement (load balancing).....365
 - usage guidelines.....326
- family mpls statement.....366
 - usage guidelines.....147, 150
- family multiservice statement.....368
 - MX Series routers
 - usage guidelines.....153
 - usage guidelines.....151
 - VPLS load balancing
 - usage guidelines.....151
- family statement
 - firewall filter.....289
 - forwarding table filters.....362
 - port mirroring.....364
 - sampling.....365
- file statement
 - helpers trace options.....371
 - packet capture.....371
 - sampling.....372
 - traceoptions.....372
 - traffic sampling output
 - usage guidelines.....315, 317, 338
- filename statement.....373
- files
 - firewall log output file.....211
 - logging information output file.....317, 338
 - traffic sampling output files.....315
 - var/log/sampled file.....317, 338
 - var/tmp/sampled.pkts file.....315
- files statement
 - packet capture.....374
 - sampling.....374
 - usage guidelines.....315
- filter statement
 - firewall.....290
 - forwarding table.....375
 - VPLS.....375
- filter-specific statement.....291
 - usage guidelines.....263
- filters
 - interface-specific counters.....216
- firewall filter
 - match conditions
 - VPLS.....193
- firewall filter lists
 - overview.....219
- firewall filters
 - actions.....174, 208
 - applying.....215, 250
 - architecture6, 9
 - comparison with routing policies9, 12
 - configuration statements.....178, 287
 - evaluation.....182
 - example filter definitions.....218, 223, 282
 - flow, packets4
 - in traffic sampling.....312
 - input lists.....219
 - log output file.....211
 - match conditions.....174, 183, 202
 - circuit cross-connects (CCC).....191
 - class-based.....201
 - IPv6.....188
 - MPLS.....192
 - protocol-independent.....191
 - match conditions,
 - IPv4.....183
 - match conditions, Layer 2 bridging (MX Series routers).....197
 - output lists.....219
 - overview.....173, 255
 - physical interface filters.....295
 - policing.....257
 - protocol families.....174
 - purpose.....9
 - service filters.....231
 - show firewall filters command.....211
 - simple filters.....232
 - standard
 - configuring.....179
 - system logging.....252
 - testing packet protocols.....200
- firewall log output file.....211
- firewall statement.....291
 - usage guidelines.....178
- flood statement.....376
 - usage guidelines.....330
- flooding
 - IS-IS and OSPF.....20
- flow aggregation.....317
- flow control actions.....47, 49
- flow-active-timeout statement.....376
 - accounting
 - usage guidelines.....330
 - sampling
 - usage guidelines.....315
- flow-export-destination statement.....377
 - usage guidelines.....332
- flow-inactive-timeout statement.....377
 - accounting
 - usage guidelines.....330
 - sampling
 - usage guidelines.....315
- font conventions.....xxix
- forwarding table
 - filters.....250, 329
 - policy, routing
 - applying.....29, 64
- forwarding-class
 - firewall filter action modifier.....208

forwarding-options statement.....	378
usage guidelines.....	325
from statement.....	167
firewall filters	
usage guidelines.....	182, 183
policy, routing	
omitting.....	59
usage guidelines.....	41
FTP traffic, sampling.....	323

G

group statement	
DHCP snooping.....	378

H

hash-key statement.....	379
usage guidelines.....	326
hash-seed statement	
load balancing.....	410, 411
helpers statement.....	382
hierarchical policer.....	272
hierarchical-policer statement	
usage guidelines.....	272

I

icmp-code (firewall filter match condition).....	187
icmp-type (firewall filter match condition).....	200
icons defined, notice.....	xxviii
if-exceeding statement.....	292
if-route-exists statement.....	160
import routing policies	
applying.....	27, 57, 60, 165
overview.....	17
import statement.....	165
policy, routing	
usage guidelines.....	27, 57
indexed-next-hop statement.....	384
input statement.....	385
firewall filters	
usage guidelines.....	215, 250
forwarding table.....	386
port mirroring.....	386
sampling.....	387
traffic sampling.....	385
usage guidelines.....	313
usage guidelines.....	329
install-nexthop lsp (routing policy action).....	51
instance (routing policy match condition).....	43
instance statement	
port mirroring.....	388
instance-name.inet.0 routing table.....	19
interface (routing policy match condition).....	43
interface policers.....	269, 270

interface set.....	269
interface statement	
accounting or sampling.....	389
BOOTP.....	390
DNS or TFTP packet forwarding or relay	
agent.....	391
monitoring.....	392
next-hop group.....	392
port mirroring.....	393
snooping.....	391
usage guidelines.....	335
interface-set statement.....	293
usage guidelines.....	269
interface-specific statement.....	293
usage guidelines.....	216
invert-match statement	
usage guidelines.....	111
IP addresses	
sampling traffic from single IP addresses.....	322
ipsec-sa	
firewall filter action modifier.....	208
IPv4	
firewall filter match conditions.....	183
IPv6	
firewall filter match conditions.....	188
IPv6 accounting, configuring.....	330
IS-IS	
policy, routing	
applying.....	28

J

joins, PIM	
rejecting.....	129

L

layer-3 statement	
load balancing	
usage guidelines.....	153
layer-3-only statement	
usage guidelines.....	151
layer-4 statement	
load balancing	
usage guidelines.....	153
LDP	
policy, routing	
applying.....	28
level (routing policy match condition).....	44
load balancing	
Ethernet pseudowires.....	150
per-flow.....	333, 335
per-packet.....	144
IPv4.....	145
mpls.....	147
per-prefix.....	334

VPLS	
M120 and M320 routers.....	151
VPLs	
MX Series routers.....	153
load-balance group.....	278
load-balance statement.....	394
usage guidelines.....	334, 335
load-balance-group statement.....	294
usage guidelines.....	278
local-dump statement.....	395
usage guidelines.....	317, 319
local-preference	
policy, routing	
action.....	51
match condition.....	44
log	
firewall filter action modifier.....	208
log output	
firewall filters.....	211
traffic sampling.....	317, 338
logical routers <i>See</i> logical systems	
logical systems	
configuring firewall filters.....	234
logical-bandwidth-policer statement.....	294
usage guidelines.....	277
logical-interface-policer statement.....	295
usage guidelines.....	271, 281
longer route list match type.....	120, 123
loss-priority	
firewall filter action modifier.....	208

M

manuals	
comments on.....	xxx
match conditions	
firewall filters	
address filter.....	203
bit-field.....	206
from statement.....	183
overview.....	174, 183
VPLS.....	193
policy, routing.....	24, 41
max-packets-per-second statement.....	395
maximum-capture-size statement.....	396
maximum-hop-count statement.....	396
usage guidelines.....	335
maximum-packet-length statement.....	397
metric	
policy, routing	
action.....	52
match condition.....	44
minimum-wait-time statement.....	397
mirror-once statement.....	398
monitoring statement.....	399
usage guidelines.....	332

MPLS	
firewall filters	
match conditions.....	192
policy, routing	
applying.....	28
multicast-scoping	
policy, routing	
match condition.....	44

N

names	
policy, routing.....	40
neighbor (routing policy match condition).....	45
next policy (routing policy control action).....	49
next term	
firewall filter action.....	208
next term (routing policy control action).....	49
next-hop	
policy, routing	
action.....	53
match condition.....	45
next-hop groups.....	333
next-hop statement.....	400
next-hop groups	
usage guidelines.....	333
next-hop-group statement.....	401
usage guidelines.....	333
no-filter-check statement.....	402
no-label-1-exp statement	
usage guidelines.....	147
no-listen statement	
usage guidelines.....	335
no-local-dump statement.....	395
usage guidelines.....	317
no-stamp statement.....	422
usage guidelines.....	315
no-world-readable statement.....	427
usage guidelines.....	317
noncontiguous address filter.....	204
notice icons defined.....	xxviii

O

origin	
policy, routing	
action.....	53
match condition.....	45
orlonger route list match type.....	120, 123
OSPF	
policy, routing	
applying.....	28
output files	
firewall log output file.....	211
logging information output file.....	317, 338
traffic sampling output files.....	315

output statement.....	404
accounting.....	405
firewall filters	
usage guidelines.....	215, 250
forwarding table.....	406
monitoring.....	406
port mirroring.....	407
sampling.....	408
usage guidelines.....	329

P

packet capture.....	345
packet counter	
firewall filters.....	211
packet-capture statement.....	410
packets	
testing packet protocols.....	200
parentheses, in syntax descriptions.....	xxx
payload statement	
usage guidelines.....	151
per-flow load balancing.....	335
per-flow statement.....	410
per-prefix load balancing.....	334
per-prefix statement.....	411
physical-interface-filter statement.....	295
usage guidelines.....	273
physical-interface-policer statement.....	296
usage guidelines.....	273
PIM	
multicast traffic joins, rejecting.....	129
policy, routing	
applying.....	28
pipe ()	
bit-field logical operator.....	207
plus sign (+), bit-field logical operator.....	207
policer	
firewall filter action modifier.....	208
policer statement.....	297
firewall	
usage guidelines.....	261
policers	
example configurations.....	282
interface.....	269, 270
physical interface.....	296
configuring.....	273
policer action portion.....	261
policy (routing policy match condition).....	45
policy framework	
architecture	6
comparison of policies	9
firewall filters.....	3
overview.....	3
policy, routing.....	3

policy, routing	
access and access-internal routes.....	74
actions.....	25, 47, 51, 52, 57
applying.....	57, 164, 166
overview.....	26
architecture	6
AS path regular	
expressions.....	97, 104, 155, 156, 157
BGP damping parameters.....	161
chains	
applying.....	57
evaluation.....	30
communities.....	104, 158
comparison with firewall filters	9
configuring.....	36, 37, 39, 72
overview	23, 29
default policies and actions.....	20
evaluation	29
export policies.....	27, 163
flow, routing information	4
framework.....	16
import policies.....	17, 27, 64, 165
match conditions.....	24, 41, 47
multiple policies	
applying.....	57
evaluation.....	30
overview.....	15, 16
policy expressions	59, 64
preferences, modifying.....	53
prefix list	117, 169
prefix list filter.....	120, 170
purpose.....	9
rejecting PIM multicast traffic joins.....	129
route lists.....	121, 129
source prefixes, group.....	80
subroutines	31, 130, 134
terms.....	26, 40
testing.....	33
uses for.....	22, 23
policy-options statement.....	166
usage guidelines.....	35
policy-statement statement.....	167
from statement.....	41
then statement.....	47
to statement.....	41
usage guidelines.....	40
port (firewall filter match condition).....	200
port mirroring.....	341
mirror-once statement.....	398
multiple instances.....	388
port statement.....	412
usage guidelines.....	317
port-mirror	
firewall filter action modifier.....	208
port-mirroring statement.....	413
usage guidelines.....	341

precedence (firewall filter match condition).....187

preferences

- modifying
- with routing policies.....53

policy, routing

- action.....53
- match condition.....45

prefix list117, 169

prefix list filter.....170

prefix-action statement.....298

- usage guidelines.....264

prefix-length-range match type.....123

prefix-list (routing policy match condition).....46

prefix-list statement.....169

- usage guidelines.....118, 206

prefix-list-filter statement.....170

premium statement

- usage guidelines.....272

protocols

- applying policies.....26
- firewall filter match condition.....187
- match condition

 - policy, routing.....46

- routing

 - applying policies.....64

- testing packet protocols.....200

R

rate statement.....414

- usage guidelines.....313

reject

- firewall filter action.....208
- policy, routing

 - control action.....49

relay agents

- DHCP and BOOTP.....335

relay agents, DHCP

- extended

 - access and access-internal routes.....74

rib (routing policy match condition).....46

RIP

- policy, routing

 - applying.....28

route lists.....121

route recording.....317

route-accounting statement.....415

- usage guidelines.....330

route-filter (routing policy match condition).....46

routers

- DHCP relay agents.....335

routing policies

- dynamic

 - configuring.....67

- dynamic database.....66

routing policy *See* policy, routing

routing tables

- instance-name.init.0.....19

routing-instance

- firewall filter action.....208

routing-instance statement

- usage guidelines.....335

RPF

- firewall log and count.....212

run-length statement.....415

- usage guidelines.....313

S

sample

- firewall filter action modifier.....208

sample (firewall filter action).....312

sampled file.....317, 338

sampled.pkts file.....315

sampling statement.....416

- usage guidelines.....311, 312

server statement

- DHCP and BOOTP service419
- DNS and TFTP service419
- usage guidelines.....335

service filters.....231

service-filter statement

- firewall.....299

show chassis hardware command

- usage guidelines.....309

show firewall filter command.....211

show interfaces policers command.....270

show log command.....212

show policers command.....259

show policy damping command.....143

- usage guidelines.....139

show route detail command

- usage guidelines.....139

show route receive-protocol command

- usage guidelines.....47

simple filters.....232

simple-filter statement

- firewall.....300
- usage guidelines.....232

single-rate statement

- usage guidelines.....278

size statement

- packet capture.....421
- sampling.....421

snooping, DHCP.....340

SONET interfaces

- sampling.....321

source class usage.....54, 80

source-address (firewall filter match condition).....183, 188

source-address-filter (routing policy match condition).....47

source-class (routing policy action).....	54
source-port (firewall filter match condition).....	183, 188, 200
stamp option.....	316
stamp statement.....	422
usage guidelines.....	315
subroutines	31, 130
support, technical <i>See</i> technical support	
syntax conventions.....	xxix
syslog	
firewall filter action modifier.....	208

T

table statement.....	160
tag	
policy, routing	
action.....	54
technical support	
contacting JTAC.....	xxx
term statement	
firewall.....	301
policy	
usage guidelines.....	40
terms	
policy, routing.....	26, 40
test policy command	33
TFTP	
packet forwarding.....	337
requests, disabling recognition.....	337
tftp statement.....	422
then statement.....	167
firewall filters	
usage guidelines.....	182, 208
policy, routing	
usage guidelines.....	47
three-color-policer statement.....	304
usage guidelines.....	278
through route list match type.....	123
timestamp option.....	316
to statement.....	167
usage guidelines.....	41
topology	
firewall filter action.....	208
trace (policy tracing action).....	47, 56
traceoptions statement	
DNS and TFTP packet forwarding.....	424
port mirroring and traffic sampling.....	426
usage guidelines.....	327
tracing actions.....	47, 56
traffic	
accounting.....	330
forwarding	
configuration statements.....	325
overview.....	325

monitoring.....	332
sampling	
configuration statements.....	311, 349
disabling.....	315, 359
DNS and TFTP packet forwarding.....	337
example configurations.....	321
flow aggregation.....	317
FTP traffic.....	323
logging information output	
file.....	317, 329, 338
output files.....	315
overview.....	311
run-length parameter.....	313
sampling rate parameter.....	313
show log command.....	212
SONET interfaces.....	321
traffic from single IP addresses.....	322

traffic sampling	
configuring.....	312
traffic-class (firewall filter match condition).....	183, 188
tricolor marking policer.....	278
interface policer.....	280
two-rate statement	
usage guidelines.....	278

U

unicast RPF.....	224
upto route list match type.....	123

V

var/log/sampled file.....	317, 329, 338
var/tmp/sampled.pkts file.....	315
version statement.....	426
usage guidelines.....	317
version9 statement.....	427
usage guidelines.....	320
virtual-channel statement.....	305
VPLS load balancing	
M120 and M320 routers.....	151
MX Series routers.....	153

W

world-readable statement.....	427
usage guidelines.....	317

Index of Statements and Commands

A

accounting statement.....	350
accounting-profile statement.....	287
action statement.....	288
aggregation statement.....	351
apply-path statement.....	155
as-path statement.....	156
as-path-group statement.....	157
autonomous-system-type statement.....	352

B

bootp statement.....	353
----------------------	-----

C

cflowd statement.....	354
client-response-ttl statement.....	358
community statement.....	158
condition statement.....	160

D

damping statement.....	161
description statement	
helper service or interface.....	358
dhcp-relay statement	
DHCP snooping.....	359
disable statement	
packet capture.....	359
sampling.....	359
domain statement.....	360
dynamic-db statement.....	162

E

ether-pseudowire statement.....	366
export statement.....	163
export-format statement.....	360

F

family inet statement (load balancing).....	365
family mpls statement.....	366

family multiservice statement.....	368
family statement	
firewall filter.....	289
forwarding table filters.....	362
port mirroring.....	364
sampling.....	365
file statement	
helpers trace options.....	371
packet capture.....	371
sampling.....	372
traceoptions.....	372
filename statement.....	373
files statement	
packet capture.....	374
sampling.....	374
filter statement	
firewall.....	290
forwarding table.....	375
VPLS.....	375
filter-specific statement.....	291
firewall statement.....	291
flood statement.....	376
flow-active-timeout statement.....	376
flow-export-destination statement.....	377
flow-inactive-timeout statement.....	377
forwarding-options statement.....	378
from statement.....	167

G

group statement	
DHCP snooping.....	378

H

hash-key statement.....	379
helpers statement.....	382

I

if-exceeding statement.....	292
if-route-exists statement.....	160
import statement.....	165
indexed-next-hop statement.....	384

input statement.....	385
forwarding table.....	386
port mirroring.....	386
sampling.....	387
instance statement	
port mirroring.....	388
interface statement	
accounting or sampling.....	389
BOOTP.....	390
DNS or TFTP packet forwarding or relay	
agent.....	391
monitoring.....	392
next-hop group.....	392
port mirroring.....	393
snooping.....	391
interface-set statement.....	293
interface-specific statement.....	293

L

load-balance statement.....	394
load-balance-group statement.....	294
local-dump statement.....	395
logical-bandwidth-policer statement.....	294
logical-interface-policer statement.....	295

M

max-packets-per-second statement.....	395
maximum-capture-size statement.....	396
maximum-hop-count statement.....	396
minimum-wait-time statement.....	397
mirror-once statement.....	398
monitoring statement.....	399

N

next-hop statement.....	400
next-hop-group statement.....	401
no-filter-check statement.....	402
no-stamp statement.....	422
no-world-readable statement.....	427

O

output statement.....	404
accounting.....	405
forwarding table.....	406
monitoring.....	406
port mirroring.....	407
sampling.....	408

P

packet-capture statement.....	410
per-flow statement.....	410

per-prefix statement.....	411
physical-interface-filter statement.....	295
physical-interface-policer statement.....	296
policer statement.....	297
policy-options statement.....	166
policy-statement statement.....	167
port statement.....	412
port-mirroring statement.....	413
prefix-action statement.....	298
prefix-list statement.....	169
prefix-list-filter statement.....	170

R

rate statement.....	414
route-accounting statement.....	415
run-length statement.....	415

S

sampling statement.....	416
server statement	
DHCP and BOOTP service	419
DNS and TFTP service	419
service-filter statement	
firewall.....	299
show firewall filter command.....	211
show interfaces policers command.....	270
show log command.....	212
show policers command.....	259
show policy damping command	143
simple-filter statement	
firewall.....	300
size statement	
packet capture.....	421
sampling.....	421

T

table statement.....	160
term statement	
firewall.....	301
test policy command	33
tftp statement.....	422
then statement.....	167
three-color-policer statement.....	304
to statement.....	167
traceoptions statement	
DNS and TFTP packet forwarding.....	424
port mirroring and traffic sampling.....	426

V

version statement.....	426
version9 statement.....	427
virtual-channel statement.....	305

W

world-readable statement.....427

